VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

CHCU VÍNO! – MOBILNÍ APLIKACE PRO KOMUNIKACI ZÁKAZNÍKA S VINAŘEM

CHCU VÍNO! – MOBILE APP FOR COMMUNICATION OF A CUSTOMER WITH A WINE MAKER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MARTIN ADAMEC**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/20509/2017/xadame36

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Adamec Martin, Bc.**
Obor: Informační systémy
Téma: **Chcu víno! - mobilní aplikace pro komunikaci zákazníka s vinařem**
Chcu víno! - Mobile App for Communication of a Customer with a Wine Maker
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s problematikou návrhu a vývoje mobilních aplikací; zaměřte se na platformu Android.
2. Vyhledejte a analyzujte existující aplikace řešící podobnou úlohu.
3. Navrhněte a prototypujte způsob interakce s aplikací a jednotlivé prvky uživatelského rozhraní.
4. Navrhněte a implementujte řešenou aplikaci - jak rozhraní pro vinaře, tak rozhraní pro zákazníka.
5. Testujte vytvořené aplikace na uživatelích a iterativně je vylepšujte.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN-13: 978-0321965516
- Android Developers: https://developer.android.com/index.html
- Susan M. Weinschenk: 100 věcí, které by měl každý designér vědět o lidech, Computer Press, Brno 2012

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3, značné rozpracování bodu 4.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Herout Adam, prof. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem této práce je vytvořit mobilní aplikaci pro operační systém Android, která propojí především drobné vinaře se zákazníky. V aplikaci si potenciální zákazník zadá oblast hledání a zprávu pro vinaře. Na základě této poptávky se rozešlou zprávy vinařům v dané oblasti. Přijmutím poptávky vinařem dostane zákazník informaci, že je dostupný. Případné upřesnění poptávky či nabídky lze uskutečnit pomocí soukromých zpráv v aplikaci. Aplikace je vytvořena s důrazem na jednoduchost použití a respektuje Material Design. Data jsou ukládána pomocí technologie Firebase od Google, notifikace jsou zasílány pomocí služby Firebase Cloud Messaging. S ohledem na jednoduchost jsou výsledkem dvě aplikace, první Mám víno! je určena pro vinaře. Druhá aplikace Chcu víno! slouží výhradně pro zákazníky, kteří pomocí ní vyhledají vinaře.

Abstract

The aim of this thesis is to create a mobile application for Android that will connect small wineries with their customers. The result is two applications, the first of which is for wineries and the second one for customers, who can search wineries by simple criteria, save them as favorites, make notes and much more. The applications respects Material Design rules. Google Firebase is used for work with data and Firebase Cloud Messaging handles notifications between devices.

Klíčová slova

víno, vinařství, Firebase, Firestore, Butter Knife, Android, FCM, mobilní aplikace, Material Design

Keywords

wine, winery, Firebase, Firestore, Butter Knife, Android, FCM, mobile application, Material Design

Citace

ADAMEC, Martin. *Chcu víno! – mobilní aplikace pro komunikaci zákazníka s vinařem.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Chcu víno! – mobilní aplikace pro komunikaci zákazníka s vinařem

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana prof. Ing. Adama Herouta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Adamec
22. května 2018

Poděkování

Chtěl bych poděkovat vedoucímu mé diplomové práce panu prof. Ing. Adamu Heroutovi, Ph.D. za pomoc a cenné rady při konzultacích.

Obsah

Kapitola 1

Úvod

Následující text seznámí čtenáře s mobilní aplikací, která má za cíl udělat propojení mezi dvěma aktuálními trendy. Žijeme v době, kdy mobilní zařízení zasahují do našeho života každým dnem více a více. Lidé je využívají nejen ke vzájemné komunikaci, ale především pro vyhledávání informací o produktech, událostech, firmách atd.

Druhým trendem je čím dál více se rozšiřující skupina lidí, která vyhledává malé (nekomerční) vinaře a jejich produkty, ať už se jedná o víno, burčák či jiné výrobky. Hlavním důvodem je stále stoupající spotřeba vína v České republice[1] a především rostoucí obliba kvalitních vín [11], která velmi často produkují právě menší vinaři.

Vyvíjená aplikace se zaměřuje na situaci, kdy se potenciální zákazník nachází ve vinařské lokalitě a rád by ochutnal lokální vína. Pomocí aplikace zašle poptávku okolním vinařům, dle zvolených parametrů. Vinař zvolí, jestli je dostupný či nikoliv. Zákazník obdrží nazpět seznam vinařů, kteří jsou k dispozici. Z tohoto seznamu si zákazník může vybrat vinaře, kteří ho oslovili a dochází k přímé interakci pomocí soukromých zpráv v aplikaci, kde si domluví detaily setkání. Podrobnější popis a analýza aplikace je v kapitole 2.

Ovšem, aby aplikace nebyla pouze jednoúčelová, lze zobrazit seznam vinařů dle vzdálenosti a vyhledávat v něm. Pokud vinař zákazníka zaujme, může zahájit vzájemnou komunikaci prostřednictvím soukromých zpráv. Aplikace nabízí i další funkce, díky nimž se může stát vhodnou pomůckou pro milovníky vína.

Komunikace probíhá za pomoci notifikací, které upozorňují uživatele na novou zprávu, poptávku či reakci na poptávku. Technické specifikaci projektu se věnuje kapitola 4, s kterou souvisí kapitola 3 o platformě *Android*.

Výsledkem práce jsou dvě mobilní aplikace pro systém *Android*. Jednak *Chcu víno!* určená zákazníkům a *Mám víno!* pro vinaře. Rozdělením byla snaha o zachování jednoduchosti grafického uživatelského rozhraní aplikace. Informace zobrazené na hlavní obrazovce pro vinaře a zákazníky se liší. Vznikem dvou aplikací vznikla možnost být současně zákazník i vinař a to pod odděleným účtem.

Při implementaci (kapitola 5) bylo třeba vyřešit několik problémů a zvolit vhodná řešení s ohledem na znovupoužitelnost kódu pro obě vyvíjené aplikace.

Důraz byl kladen na iterativní testování aplikací. První iterace byla provedena po vytvoření skic grafického uživatelského rozhraní. Následně probíhala iterativní implementace a snaha o nalezení nelogických řešení v aplikacích. Všechny testy a jejich výsledky jsou uvedeny v kapitole 6.

[1] Spotřeba a produkce vína v ČR – http://multimedia.ctk.cz/grafika/index?documentID=8476125

Kapitola 2

Specifikace a analýza řešených aplikací

Hlavním účelem aplikace je nalezení nových zákazníků pro drobné vinaře a vinařství (dále jen „vinaře“). Dalším cílem je umožnit zákazníkům poznat nové chutě a odrůdy vín, nebo jiné produkty vinařů. Námět tvorby vznikl z osobních zkušeností: má rodina vlastní vinohrady a produkuje víno již několik generací, naším cílem není prodávat ve velkoobchodech, víno je pro naši potřebu, ale rádi se podělíme se známými, kamarády nebo náhodnými turisty, respektive malými zákazníky.

Současně je na vzestupu turistika za vínem, tedy roste počet potenciálních zákazníků. V aktuální době nemá zákazník dostupný prostředek, jak zjistit, jestli je nějaký vinař k dispozici nebo ne. Raději zakoupí víno ve vinárně či obchodě, ale tím se může ochudit o autentičnost chutí vín malých vinařů. Právě tuto mezeru na trhu se snaží vyřešit aplikace, která je v tomto textu popisována.

2.1 Popis základní funkčnosti aplikace

Základní principem aplikace je vytvoření vyhledávání pro zákazníka. Vyhledávání obsahuje informace o prohledávané oblasti a zprávu pro vinaře.

Vinaři ve zvolené oblasti dostanou zaslanou zprávu (poptávku), pomocí notifikace. Záleží pouze na vinaři, jestli má o poptávku zájem, může ji přijmout nebo zamítnout, případně připojit soukromou zprávu.

Zákazník obdrží seznam vinařů, kteří přijali poptávku, tedy ví, kdo je k dispozici a může se za nimi stavit pro víno. Pro přesnější domluvu mezi zákazníkem a vinařem slouží soukromé zprávy. Komunikace využívá notifikací pro upozornění uživatele na událost.

Cílem není vytvoření aplikace, která vytváří komunitu uživatelů, ale inteligentního nástroje, sloužícího pro soukromé účely jednotlivce.

Potenciál a nástrahy aplikace

Z pohledu využitelnosti a možného potenciálu aplikace bylo třeba si uvědomit několik základních věcí, respektive problémů.

- Aplikace bude využívána zákazníky **ojediněle**. Zákazník nepotřebuje pokaždé zkusit nového vinaře, typicky si najde jednoho či více, ke kterým se vrací. Aplikaci může

využít především na dovolené v oblasti, kde nezná vinaře a má chuť objevit něco nového.

- **Omezené časové období**, kdy bude využívána. Podobně jako v předchozím bodě, na dovolenou do vinařské oblasti se jezdí především v teplých měsících a zejména v období letních prázdnin.
- **Zavedení** aplikace do dané oblasti **potřebuje čas a spolupráci** lokálních podniků. Nejlepší propagace by mohla být pomocí lokálních poskytovatelů ubytování nebo občerstvení. To lze realizovat nejlépe pomocí lokální osoby, která majitele zařízení zná.
- Aplikace si současně musí získat důvěryhodnost a především ukázat **přínos vinařům**.
- Představení aplikace jednotlivým vinařům je nejlépe realizovat osobně. Znovu je potřeba osoby žijící v dané oblasti.

Výše zmíněné faktory předurčují, že zavedení aplikace bude náročné především na lidské zdroje. S těmito faktory je třeba počítat a zohlednit v návrhu aplikace, který je popsán v kapitole 4.

Při analýze bylo zjištěno, že neexistuje přímá konkurence (viz kapitola 2.3). Tím je otevřena možná expanze do zahraničí, především země jako Francie, Španělsko, Portugalsko, Rakousko či Itálie představují zajímavou výzvu.

2.1.1 Funkce aplikace

Hlavním účelem aplikace je dostat zákazníka v co nejjednodušším procesu k vinaři. Současně aplikace má být plnohodnotnou pomůckou pro zájemce o víno, tedy je třeba přidat i další funkcionalitu:

- **Profil uživatele** – sloužící především pro rychlé získání kontaktních informací, jako jsou souřadnice vinného sklepa, telefon apod.
- **Seznam vinařů** – vinaři seřazení podle vzdálenosti od zákazníka s možným filtrováním a vyhledáváním.
- **Soukromé hodnocení uživatelů** – umožnit napsat si soukromou poznámku k uživateli. Např.: víno mi chutnalo a nebo naopak, nebyl ve sklepě, i když avizoval, že je.
- **Soukromé zprávy** – umožnit kontaktovat vinaře v privátní konverzaci.

Monetizace aplikace

Jelikož mám v úmyslu pokračovat ve vývoji aplikace dále, bylo třeba zvážit i možnosti monetizace aplikace. Je zapotřebí se vyhnout úvahám o placeném účtu pro zákazníky. Dle mého názoru by to bylo kontraproduktivní.

Druhou možností je vytvořit placenou verzi pro vinaře. Při hledání možností bylo zohledněno především ponechání plnohodnosti aplikace i pro neplacené vinaře. Nabízí se poskytnout platícím uživatelům benefity jako jsou:

- prioritní oznámení o poptávce zákazníka (ostatní vinaři dostanou oznámení se zpožděním),

- tzv. „topování“ v seznamu vinařství, tzn. vinaři s placenou verzí budou zobrazování přednostně, i třeba s jinou grafikou,
- rozšířená funkcionalita, jako možnost přidat informace o aktuálních vínech, galerie obrázků, ubytování, objednání rautu a degustací.

2.2 Cílová skupina

Cílovou skupinou aplikace jsou jednak všichni vyznavači dobrého vína, kteří rádi objeví odlišnosti chutí vín jednotlivých regionů. Uživatel však nemusí být pouze znalec vína, ale i obyčejný člověk, který je ve vinařské oblasti na výletě nebo dovolené a rád ochutná lokální produkty.

Je známo, že především mladší lidé využívají mobilní telefony, aplikace a internet pro nalezení informací. Lze tedy předpokládat, že tato skupina bude nejčastější uživatel aplikace. Proto je dobré využít moderní navigační prvky a vhodně zvolené ikony, které pomohou uživateli v orientaci.

Druhá skupina uživatelů jsou vinaři, především malovinaři. Tato skupina většinou nechce využívat placených aplikací nebo propagování pomocí reklamy, jelikož pro velikost jejich produkce se to nevyplatí. Díky aplikaci *Chcu víno!* mohou zdarma a snadno získat nové jednorázové i dlouhodobé zákazníky bez většího úsilí.

2.3 Existující řešení

Existuje mnoho aplikací, které pracují s polohou a vyhledávají okolní body. Jako příklad lze využít navigaci, která uživatele umí dovést na nejbližší čerpací stanici. Dalším všeobecně známým příkladem, jsou mobilní aplikace vyhledávající nejbližší restauraci (například s poledním menu).

Zasílání notifikací jiným uživatelům, tedy druhý základní kámen vyvíjené aplikace, je také hojně používané. Jako příklad mohou posloužit komunikátory: *Messanger*[1], *Viber*[2] a podobně. Komunikátor nejdříve zasílá notifikaci uživateli o nové zprávě (jedná se o rychlou akci, přenáší pouze objemově malá data). Až následně probíhá stahování celé zprávy aplikací, ať už při zobrazení, nebo automaticky na pozadí.

Aplikaci *Chcu víno!* lze stručně a obecně charakterizovat následovně: vyhledávání poskytovatelů služeb v okolí uživatele, kde poskytovatel potvrzuje, že je k dispozici. Při zaměření na aplikace splňující tuto charakteristiku bylo zjištěno, že jich mnoho není. Aplikace, kterou lze nazvat přímou konkurencí, nebyla nalezena.

Na kole i pěšky

Snad jedinou obdobnou aplikací je projekt *Na kole i pěšky*[3], který nabízí mobilní aplikaci pod názvem **nakoleipesky**. Aplikace obsahuje i vinaře. Vyhledání služeb probíhá na stejném principu zvolení vzdálenosti od uživatelovy polohy. Mimo poskytovatele služeb vyhledá uživateli kulturní akce v okolí, turistické a cyklistické stezky.

Na základě cílení a provedení tohoto projektu bych jej nenazval konkurencí, ale spíše možnou inspirací. Aplikace *Chcu víno!* je zaměřena na konkrétní segment trhu (vinaře)

[1] Messanger – https://messenger.com/
[2] Viber – https://www.viber.com/
[3] Na kole i pěšky – https://www.nakoleipesky.cz

a cílem je nejen nalezení vinaře zákazníkovi, ale také možnost si zaznamenávat poznámky a další, a tím se stát užitečnou aplikací milovníkům vína.

2.3.1 Aplikace s podobnou tématikou

Lze najít nespočetně mnoho aplikací se zaměřením na víno. Jedná se o aplikace jednoduché i složité, některé jsou amatérské po grafické úrovni, jiné s promyšleným designem.

Cílení těchto aplikací je různorodé, od nástrojů pro soukromé poznámky k vínům, přes vyhledávače vín či vinařství v konkrétní oblasti, až po aplikace určené pro nákup vína přes mobilní zařízení. Takovéto aplikace lze rozdělit do několika základních skupin, které se mohou překrývat.

Lokální

Zaměřené pouze na konkrétní oblast s cílem ji prezentovat. Často vinařství musí platit poplatek, aby byla součástí seznamu. Jedná se pouze o vyhledávač s rejstříkem vinařství, neprobíhá žádná interakce.

Např.: Strade del Vino di Toscana[4].

Obecné

Všeobecné aplikace, které přinášejí uživatelům znalosti (informace o jednotlivých odrůdách), vyhledávají víno dle etikety nebo poskytují záznamník dojmů – přizpůsobený formulář pro hodnocení vín. Aplikace většinou neposkytují informace o konkrétních vinařstvích.

Např.: S Víno – seznam, hodnocení, sklepy[5].

Soukromé

Slouží jako soukromá prezentace konkrétního vinařství. Obsahuje více informací, může být doplněno i o obsazenost ubytování, rozšiřující služby atd.

Např.: Vino Mrva&Stanko[6].

Jednoúčelové

Aplikace vytvořené pro podporu prezentace události (festival vína, veletrh apod.). Součástí je i karta vinařství, avšak neprobíhá žádná interakce v rámci aplikace.

Např.: Bordeaux Fête le Vin[7].

Prodejní

Prostřednictvím těchto aplikací lze naleznout víno dle preferencí, např. dle cukernatosti nebo vhodnosti ke konkrétnímu účelu, a zakoupit jej nebo zjistit, kde se ono víno prodává. Některé aplikace nabízejí možnost si napsat soukromé hodnocení, zobrazují kartu vinařství, ale stále zde chybí interakce mezi zákazníkem a vinařem.

Např.: Vivino: Wine Made Easy[8].

[4] https://play.google.com/store/apps/details?id=bbs.sdv
[5] https://play.google.com/store/apps/details?id=com.metosphere.winefree
[6] https://play.google.com/store/apps/details?id=org.mrvastanko.app
[7] https://play.google.com/store/apps/details?id=fr.nfcinteractive.bordeauxfetelevin
[8] https://play.google.com/store/apps/details?id=vivino.web.app

2.3.2 Zhodnocení existujících řešení

Existující aplikace neposkytují uživateli informaci, jestli je vinař aktuálně k dispozici nebo ne, a slouží spíše jako rejstříky statických dat. Některé aplikace vyžadují poplatek za uveřejnění.

Z hodnocení aplikací lze vypozorovat, že jsou především zaměřené na jeden konkrétní účel a chybí tam přesah. Například aplikace, která obsahuje rejstřík vín, neposkytuje uživateli možnost hodnotit vinařství či vína.

Hlavním zjištěním analýzy je, že aplikace (kromě soukromých) se nezaměřují na malá vinařství, ale především na vinaře, kteří produkují tisíce či miliony lahví ročně. Další podstatná věc je, že aplikace nezastávají funkci komunikátoru, zákazník zjistí statická data jako adresu, kontaktní údaje apod., ale interakci musí zahájit mimo aplikaci. Při návrhu (kapitola 4) aplikace *Chcu víno!* je zhodnocení existujících řešení zohledněno, aby výsledný produkt byl komplexní nástroj pro konzumenty vína.

Kapitola 3

Vývoj mobilních aplikací pro Android

Android je open-source platforma na bázi Linuxu určená hlavně pro mobilní zařízení. Nejčastěji je využívána pro chytré telefony a tablety bez hardwarové klávesnice a s malou obrazovkou. Její vývoj vede společnost *Google* pod hlavičkou organizace *Open Handset Alliance*. Jde o jeden z mála operačních systémů (OS), které podporují více platforem, lze ho vidět v zařízeních nejrůznějších značek. Toto ovšem přináší nevýhody, jelikož systém není optimalizovaný na konkrétní platformu. Na druhou stranu si může výrobce zařízení *Android* do značné míry upravit pro své potřeby. S každou aktualizací *Androidu* je nutné provést úpravu od konkrétního výrobce a tím se prodlužuje doba, kdy koncový uživatel obdrží aktualizaci (např.: u bezpečnostních záplat se toto prodlení může stát klíčovým).

Lze uvažovat, že výsledná aplikace bude používána především v mobilních telefonech. Je třeba brát ohled na méně výkonné telefony, které právě nejčastěji využívají tuto platformu. Budoucím uživatelům by nebylo příjemné, pokud by aplikace zatěžovala jejich přístroj (procesor) natolik, že by znemožnil běžné úkony, jako je přijmutí volání. Dalším aspektem, na který je třeba přihlédnout při tvorbě aplikace, je rozmanitost velikosti obrazovek jednotlivých zařízení. Na tento problém *Android* již myslí a nabízí možnost odlišných nastavení (velikost ikon, obrázků, rozmístění prvků, ...) pro různé velikosti obrazovek[1].

Následující text uvádí, jak *Android* pracuje, především jsou popsány moduly, které jsou důležité při vývoji této aplikace. Převážně je čerpáno z oficiální dokumentace k *Androidu* [8] a knihy *Vývoj aplikací pro Android* [15].

3.1 Verze Androidu

Velmi důležitým krokem při započetí tvorby mobilní aplikace pro operační systém *Android* je zvolení verze OS. *Android* má číselné označení svých verzí. Ty důležité mají své unikátní (kódové) označení názvem zákusku. Ke každé verzi je také vázána úroveň API, kterou je třeba při vývoji respektovat. Výhodou je poměrně dlouhodobá (s ohledem na průměrnou životnost mobilního zařízení) kompatibilita (avšak ne zpětná).

Dle oficiálních statistik [4] je aktuálně s 28,6 % nejvíce využívána verze **6.0**, *Marshmallow* (API úroveň 23). Z předešlých verzí tvoří důležitou součást také zařízení s OS verze **5.0** a **5.1** s kódovým označením *Lollipop*, které dohromady představují 25,1 %. Starší

[1] https://developer.android.com/guide/practices/screens_support.html

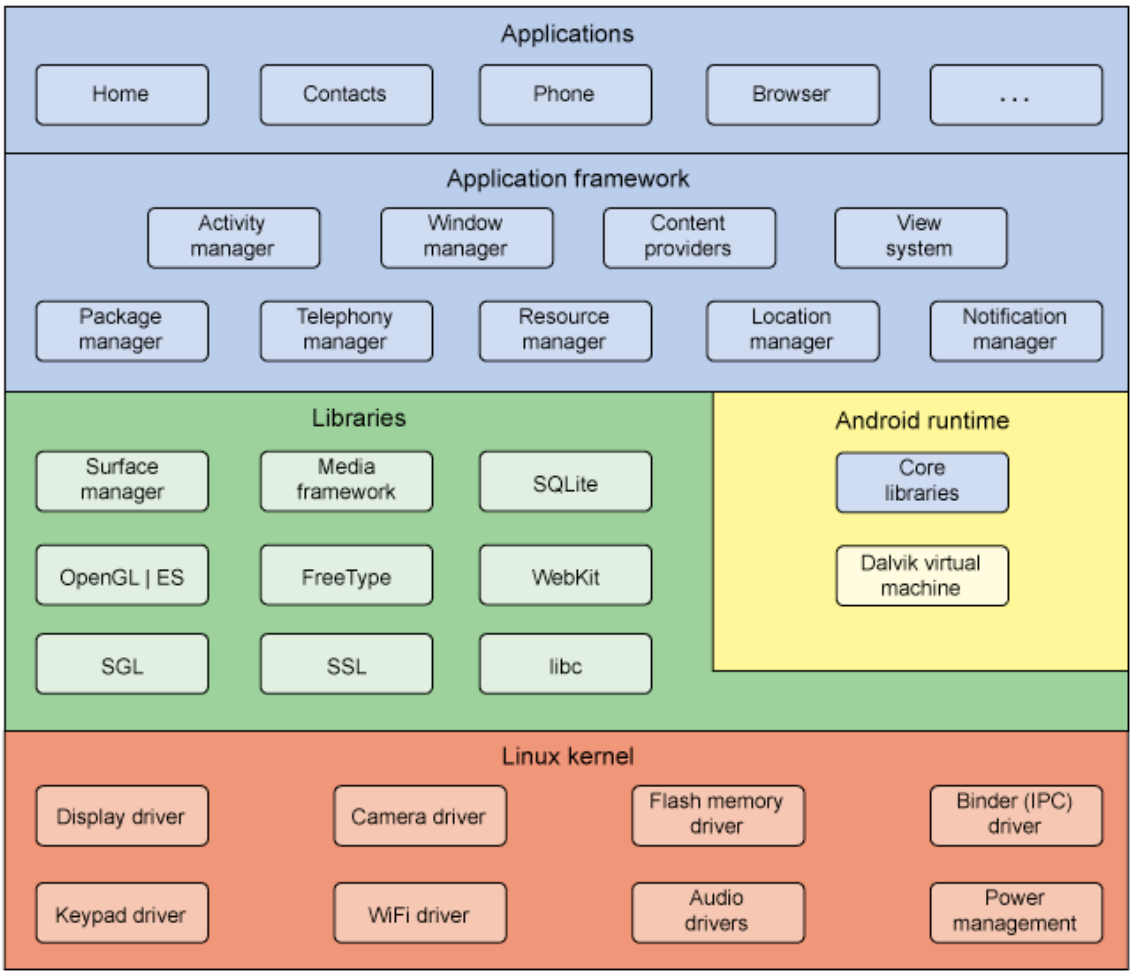
verze již rychle ztrácí své zastoupení, například verze **4.4** (*KitKat*) během posledního roku ztratila více jak polovinu uživatelů.

Pro vytvářenou aplikaci byla zvolena verze **5.0** API úroveň 21. Nadcházející text se vztahuje právě k této verzi, pokud není explicitně uvedeno jinak. Počet uživatelů, kteří využívají tuto nebo vyšší verzi, je v současnosti 80,7 % [4]. Za nejvíce důležitou změnu v této verzi *Androidu* lze považovat podporu *Material Design* [10], která je od této verze doporučována pro vytváření aplikací.

3.2 Architektura

Nezbytnou znalostí pro vývoj aplikace je porozumění architektuře *Androidu*, tedy rolím jednotlivých vrstev a jak mezi sebou komunikují.

Na obrázku 3.1 je vyobrazeno schéma OS, základní funkce jednotlivých vrstev jsou popsány níže, v pořadí od nejnižších vrstev [15].



Obrázek 3.1: Schéma architektury OS Android [1]

- **Jádro (Linux Kernel)**: nejnižší vrstva architektury je upravené Linuxové jádro. Je přizpůsobeno pro mobilní zařízení. Slouží primárně pro přímou interakci s hardwarem zařízení, čímž tvoří abstrakci od hardwaru pro vyšší vrstvy.

Zabezpečuje správu paměti, procesů a základní síťovou vrstvu. Je zde implementované také zabezpečení systému, správa napájení a další. Pomocí modulu *Binder*, který využívá sdílenou paměť, je možné sdílet údaje mezi více procesy. Procesy, které chtějí komunikovat s jiným, se musejí zaregistrovat ve službě *Service Manager*, přes níž probíhá komunikace mezi procesy.

- **Knihovny (Libraries)**: vrstva obsahuje knihovny napsány v *C/C++* a poskytující přístup aplikacím ke komponentám OS. Příkladem knihoven je *WebKit*, který je určený k renderování a zobrazování webových stránek. Využívá knihovnu *OpenGL* a práci s grafikou.

- **Android Runtime**: vrstva se stará o běh jednotlivých aplikací. Každá aplikace je samostatný proces využívající vlastní instanci virtuálního stroje, který zabezpečuje běh zkompilovaného souboru aplikace. Jedná se o obdobu *Java Virtual Machine* na klasických počítačích.

- **Aplikační framework (Aplication Framework)**: obsahuje v aplikacích opakovaně použitelný software, například ovládací prvky, ikony a podobně. Je napsán v *Java*, poskytuje základní služby systému aplikacím, jedná se tedy o nejdůležitější vrstvu pro vývoj aplikací.

  Služby poskytující aplikacím jsou **Package Manager** (modul pro správce balíčků, neboli aktuální seznam aplikací nainstalovaných v zařízení), **Window Manager** (spravuje okna, která vytváří aplikace), **View System** (spravuje společné prvky GUI[2], jako jsou ikony, tlačítka a mnohé další), **Activity Manager** (starající se o životní cyklus aplikace) a další moduly.

- **Aplikace (Applications)**: jedná se o nejvyšší vrstvu architektury operačního systému, na které se nacházejí jednotlivé aplikace.

3.3 Základní součásti aplikace

Hlavním souborem každé aplikace je speciální soubor s názvem `AndroidManifest.xml`. V tomto souboru se definují základní informace o aplikaci, jako jsou:

- název aplikace,
- potřebná přístupová oprávnění k prostředkům zařízení (např.: aktuální poloha, přístup k profilovým údajům, internetu, atd.),
- všechny dostupné aktivity,
- registrované služby (*Service*), například pro přijímání notifikací.

Aplikace pro Android jsou vybudovány na čtyřech základních pilířích realizovaných jako třídy. Pouze první představené aktivity vyžadují uživatelské rozhraní.

[2] Graphical User Interface (Grafické uživatelské rozhraní)

Aktivity (Activity)

Jedná se o třídy, které uživateli zobrazí obsah a reagují na akce. Aktivity si můžou mezi sebou předávat údaje. Role aktivit je především pro zpracování menších nebo částečných úloh, např.: vyplnění formuláře, nastavení parametrů a podobně. Zobrazuje uživatelské rozhraní a zachytává interakci uživatele.

Uspořádání aktivit je hierarchické, tedy nejprve se spustí hlavní spouštěcí aktivita, ze které je možné spouštět další aktivity.

Třídy aktivit mají předdefinované metody, které reagují na stav aktivity, které si popíšeme níže. Životní cyklus aktivit a pořadí volání předdefinovaných metod si lze prohlédnout na obrázku 3.2.

- **onCreate()**: aktivuje se při spuštění aktivity, kód metody se provádí na pozadí, aktivita je stále zastavená, neviditelná a nekomunikuje s uživatelem. Inicializuje se zde *layout* a ovládací prvky.
- **onStart()**: v momentě, kdy aktivita přechází do popředí k uživateli, volána při vytvoření nebo předchozím zastavení.
- **onResume()**: volána také při startu aktivity, jako *onStart()*, s tím rozdílem, že je volána i pokud je aktivita na pozadí a přechází na popředí.
- **onPause()**: při spuštění jiné aktivity je současná dána do pozadí. Vhodné například k automatickému uložení údajů.
- **onStop()**: při zastavení aktivity z jiného důvodu než je nedostatek paměti, například dlouhodobě nebyla zobrazena.
- **onRestart()**: pokud byla aktivita zastavena a následně ji je třeba obnovit.
- **onDestroy()**: volaná při ukončení aktivity, ať už explicitně nebo implicitně.

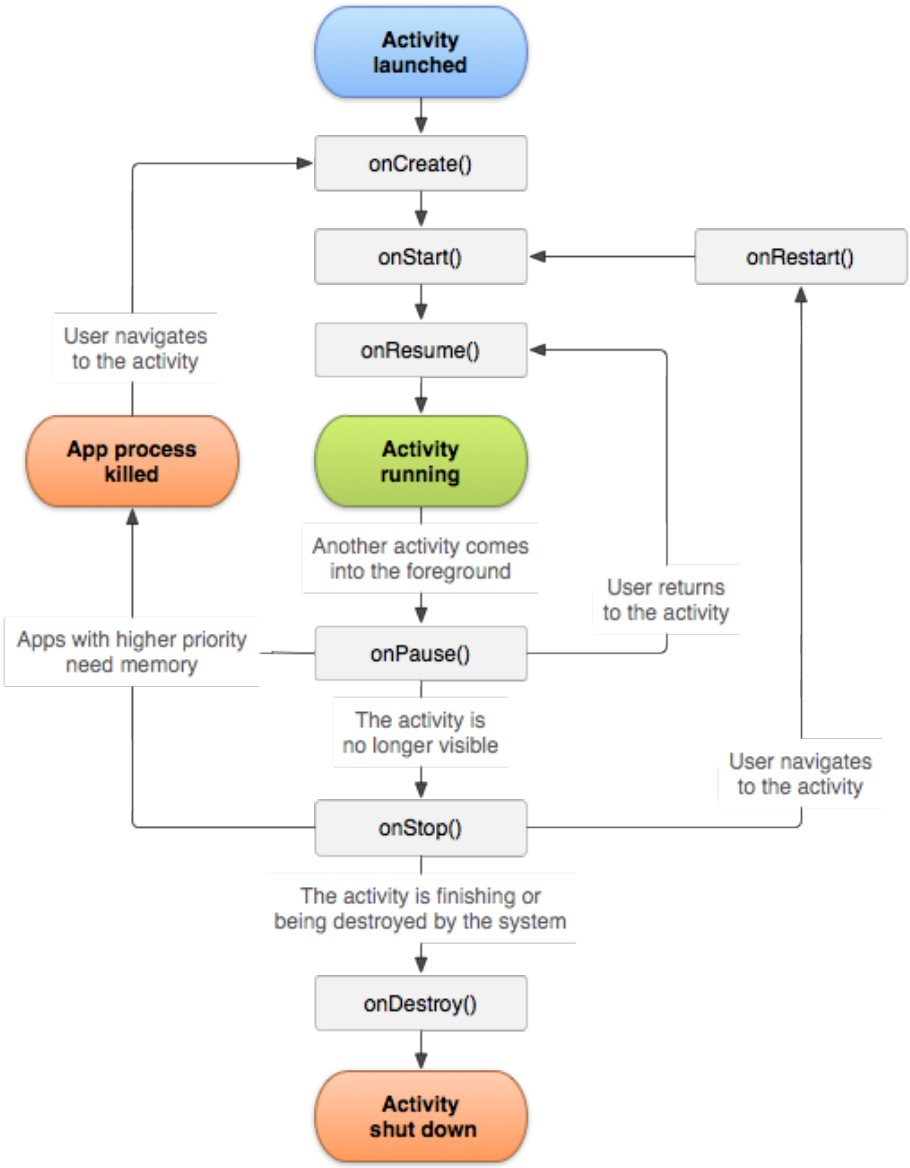
Služby (Services)

Realizují déle trvající operace a operace na pozadí. Služby mohou spolupracovat s ostatními procesy. Jedná se o instance s dlouhou životností, neukončí se ani při přepnutí do jiné aplikace nebo dokonce pokud je aplikace vypnuta. Příkladem může být přehrávač hudby pomocí služby *MediaPlaybackService.java*.

Broadcast Receivers

Jedná se o objekty pro vysílání a přijímání, poslouchají na pozadí a reagují na události. Fungování je založeno na *publish/subscribe*[3]. Vydavatelé vytvářejí záměry (*Intent*) a směřují je přes *Broadcast* do vysílání. Přijímače, které mají daný kanál zaregistrovaný, tak přijmou záměr. Příkladem je zasílání SMS zpráv.

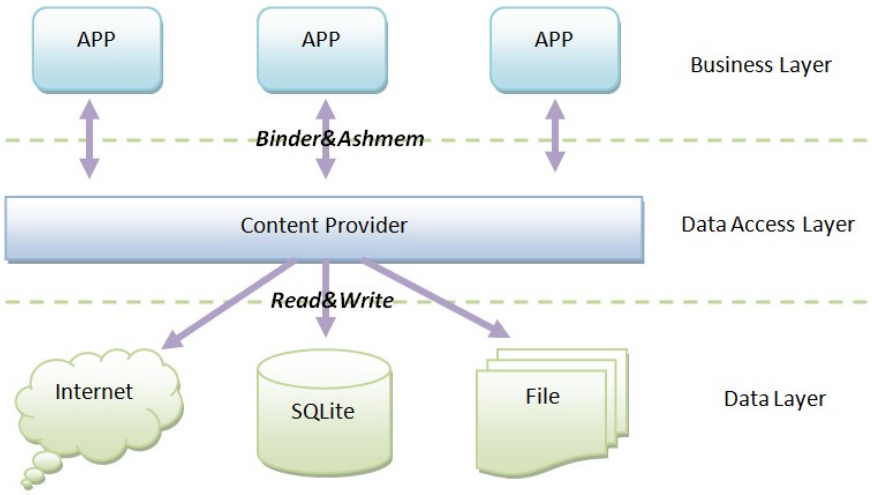
[3] https://realtimeapi.io/hub/publishsubscribe-pattern/



Obrázek 3.2: Schéma životního cyklu aktivity [13]. Při vývoji je nutné znát pořadí metod a jejich význam. Je důležité provádět akce ve správných metodách. Například v metodě *onResume()* je vhodné provést aktualizaci proměnných dat (seznam konverzací). Pokud se aplikace přesouvá na pozadí je žádoucí ukončit asynchronní akce a znovu je spustit až při pokračování aktivity.

Poskytovatelé obsahu (Content Providers)

Poskytovatelé obsahu umožňují ukládání a sdílení dat mezi více aplikacemi a procesy. Poskytovatel obsahu nabízí pomocí rozhraní práci se svými daty. Pravidla pro práci si může poskytovatel upravit dle svých potřeb a tím zůstává jediným hlavním správcem těchto dat, viz obrázek 3.3.



Obrázek 3.3: Schéma rozhraní poskytovatele obsahu [2]. Při aktualizaci dat je uvědoměn poskytovatel obsahu, který distribuuje změny do „podepsaných“ aplikací.

3.4 Záměry (Intents)

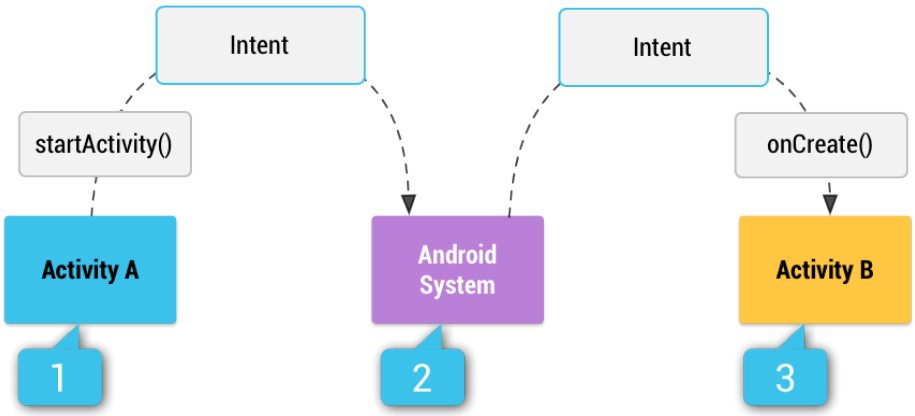
Jelikož vazby mezi aktivitami jsou slabé, tak vstupy a výstupy aktivit jsou definované pomocí záměrů. Jedná se o asynchronní zprávu, která v sobě nese informaci o požadované akci, ať už se jedná o spuštění jiné aktivity či služby. V případě, že více aktivit může vykonat daný záměr, tak systém nabídne uživateli možnost volby. Příkladem je otevření webového odkazu, který je možno realizovat více prohlížeči.

Ukázku spuštění nové aktivity pomocí záměru demonstruje obrázek 3.4.

3.5 Lokalizační a mapové služby

Mobilní přístroje přijímají signál GPS z družic, tedy můžeme vytvářet aplikace typu navigace nebo vyhledání míst (restaurace, benzínové stanice, ...) dle aktuální polohy uživatele. Systém *Android* nabízí třídu *Location*, která reprezentuje poslední získanou polohu. Třída obsahuje údaje o zeměpisné šířce, délce a čase získání. Dále lze zjistit údaje o přesnosti polohy, nadmořské výšce a podobně.

Aktuální polohu lze získat pomocí několika poskytovatelů. Základním poskytovatelem je třída využívající již zmíněný GPS signál. Další poskytovatelé pak využívají metody triangulace GSM stanic, Wi-Fi přístupových bodů nebo pomocí senzorů. Metody se od sebe rozlišují hlavně přesností a spotřebou energie. S jednotlivými poskytovateli polohy lze pracovat jednotně, jelikož mají společnou rodičovskou třídu *LocationProvider*. V případě, že



Obrázek 3.4: Demonstrace spuštění jiné aktivity pomocí záměru [7]. Aktivita vytvoří záměr, který obdrží jádro systému. Systém zpracuje záměr a vytvoří novou aktivitu, které je předáno řízení.

není potřeba explicitně využít určitého poskytovatele, lze pomocí třídy *LocationManager* získat ideálního poskytovatele polohy dle zvolených kritérií.

Nejpřesnější způsob je pomocí GPS signálu. Nevýhodou této metody je velká výpočetní náročnost, proto je potřeba polohu zjišťovat pouze, když je to nutné. Jako opatření lze využít znalost, že získávání aktuální polohy není pro každou aplikaci zvlášť, ale zajišťuje to sdílená služba. V momentě, kdy je potřeba tento údaj, lze získat poslední známou polohu, která je instancí třídy *Location*. Jednoduchým ověřením lze rozhodnout, jestli je tento údaj dostatečný (na základě času získání a přesnosti polohy), nebo je potřeba získat aktuálnější údaje.

Získání aktuální polohy v aktivitě

Pokud je v aplikaci potřeba získat aktuální polohu v aktivitě, implementuje se rozhraní *LocationListener*, které naslouchá oznámením právě od *LocationManager* v momentě, kdy dojde ke změně polohy. Při registraci k naslouchání se nastaví konkrétní poskytovatel a minimální změna vzdálenosti v metrech a času v milisekundách, kterou je nutné pro aktualizaci překonat.

Je vhodné při pozastavení aktivity toto naslouchání zastavit, aby zbytečně při změně nezatěžovalo výkon. Při pokračování aktivity lze v metodě *onResume()* opět naslouchání spustit.

Zobrazení aktuální polohy a jiných bodů v mapě

Pro lepší zobrazení informace může posloužit mapa. Systém *Android* má v sobě zabudované Google Mapy, které lze využívat pomocí *Google Maps Android API*[4]. Pro využívání API je třeba vygenerovat si API klíč a vložit jej do své aplikace.

Pomocí mapy lze uživateli ukázat lokaci určitého místa, jeho aktuální pozici nebo také nabídnout uživateli možnost vybrat bod, který je vstupem pro další pokračování aktivity. Pokud potřebuje pouze zobrazit určitou adresu či lokaci, lze samozřejmě ponechat výběr aplikace, která je schopna zobrazit mapy, na uživateli.

[4] https://developers.google.com/maps/documentation/android-api/

3.6 Ukládání dat

Stejně jako většina aplikací i *Chcu víno!* ukládá a zobrazuje uživatelská data. *Android* nabízí vývojáři několik variant jak si tato data ukládat. Jelikož aplikace pracuje nejen se svými daty, ale také s daty ostatních uživatelů aplikace, je potřeba využít i serverové úložiště a ne pouze lokální. Serverové úložiště může být na vlastním serveru nebo například pomocí služby *Firebase*. Jednotlivé varianty ukládání dat jsou popsány níže [15].

- **Shared Preferences** – jednoduché údaje typu klíč-hodnota, ukládán do vnitřní paměti zařízení
- **Internal/External Storage** – ukládání dat do vnitřní/externí paměti zařízení
- **SQLite databáze** – privátní souborová databáze, která je součástí OS, jedná se o relační databázový systém a podporuje pouze základní typy dat: *null*, binární data, textové a číselné (`INTEGER`, `REAL`) údaje
- **Firebase** (Externí) – od firmy *Google*, více viz kapitola 4.2.2
- **Vlastní** (Externí) – manipulace s daty pomocí `HTTP` požadavků, uloženo na vlastním serveru

Kapitola 4

Návrh aplikace

V následujících kapitolách je objasněn návrh aplikace. V textu jsou porovnány jednotlivé způsoby řešení a vysvětlena volba zvolených technologií. Dále je uveden návrh struktury dat, struktury aplikace a jiné technické specifikace.

Stěžejní část kapitoly je návrh grafického uživatelského rozhraní (GUI), které je zaměřeno na jednoduché a intuitivní ovládání. V návrhu je snaha vyhnout se nedostatkům existujících a podobných řešení, viz kapitola 2.3. Zároveň v kapitole 4.1.2 jsou vysvětleny důvody, proč výsledkem jsou dvě aplikace *Chcu víno!* a *Mám víno!* Postupně se dostaneme od abstraktního pojetí aplikace do konkrétního návrhu, který popisuje zvolení modelů, strukturu databáze, způsob komunikace uživatelů a také základní podobu aplikace.

4.1 Struktura aplikace

V této části jsou popsány základní principy struktury aplikace, od volby cílového operačního systému, rozštěpení aplikace až po popsání rozdělení zdrojových kódů do logických balíčků.

4.1.1 Volba operačního systému

Cílem práce je vytvořit mobilní aplikaci. Mezi první rozhodnutí patřila volba cílového operačního systému. Z analýzy jako nejlepší řešení vyplynul systém *Android*, který je detailně popsán v kapitole 3, ve které je zdůvodněno i zvolení verze *Lollipop* (API úroveň 21).

Dalším kandidátem byl operační systém *iOS* od společnosti *Apple*. Při výběru byl jako hlavní argument počet zařízení využívajících *Android* oproti *iOS* a jiným [12].

4.1.2 Rozštěpení aplikace do dvou

Z popisu aplikace (kapitola 2.1) plyne, že uživatele aplikace lze rozdělit do dvou základních skupin. V první řadě jsou to zákazníci, kteří hledají dobré víno. Na druhé straně jsou to vinaři, kteří chtějí nabídnout své produkty novým zákazníkům. Jednotlivé skupiny uživatelů mají jiná očekávání od této aplikace.

Zákazníkovi je třeba nabídnout co nejjednodušší způsob, jak nalézt vinaře v okolí, který uspokojí jeho potřeby. Vyhledání je vhodné umístit na úvodní obrazovku a intuitivně jej provést vyhledávacím procesem.

Pro vinaře je nejdůležitější jestli obdržel novou poptávku nebo zprávu od zákazníka. Proto je zbytečné mít jako první vyhledávání. Lze najít i další odlišnosti v potřebách těchto skupin uživatelů. Vzorovým příkladem je, že zákazník ocení seznam vinařství, ale vinař

jej nevyužije. Vinař by mohl uvítat seznam zákazníků, ale tím by vznikala příležitost pro nechtěnou reklamu od vinařů k zákazníkům a tím by znehodnocoval celou aplikaci. Z tohoto důvodu vinař tuto možnost nemá.

Na základě výše uvedených důvodů je aplikace rozdělená do dvou aplikací. Vznikla tak aplikace *Mám víno!*, která je určena pro vinaře, a aplikace *Chcu víno!* zůstala pro zákazníky. Aplikace mají podobné rozhraní, pro lepší odlišení má každá svou barvu a ikonu (logo), viz kapitola 4.7.

Sdílený účet vinařství

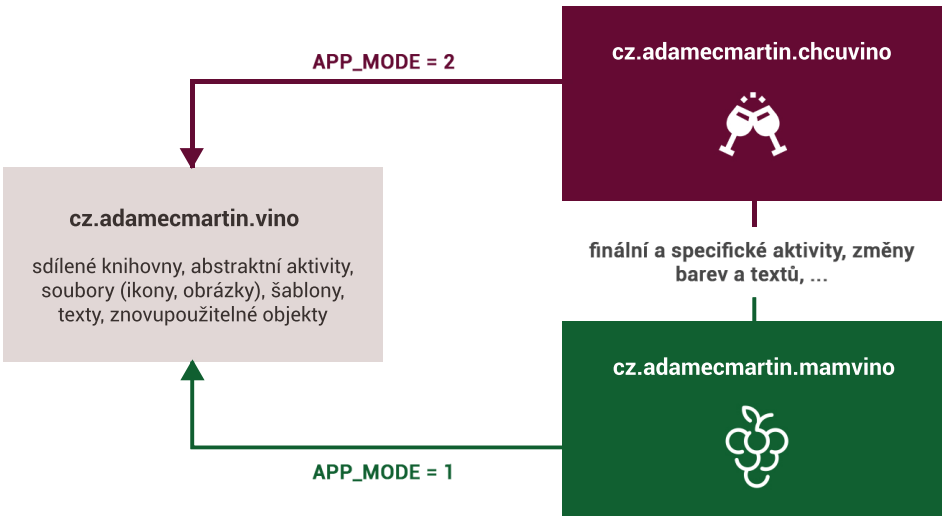
Díky vzniku dvou aplikací přibyla další výhoda pro vinaře, respektive vinařství. Obvykle je vinařství obstaráváno více jak jednou osobou, ať už členy rodiny, sousedy či kamarády. Všechny tyto osoby se mohou do aplikace *Mám víno!* přihlásit pod stejným účtem (více o způsobu autentizace v kapitole 4.3), ale v aplikaci *Chcu víno!* můžou být současně přihlášení pod svým soukromým účtem.

Při vytvoření poptávky je zaslána notifikace do všech zařízení. Zvyšuje se pravděpodobnost rychlé reakce vinařů a předání poptávky mezi členy vinařství je zautomatizované.

4.1.3 Sdílený modul aplikací

Projekt se skládá ze dvou aplikací. Velká část aplikací využívá podobnou logiku, pracuje se stejnými entitami i jednotlivé aktivity jsou často téměř stejné, např. konverzace uživatelů.

Aby nevznikal redundantní kód, obě aplikace spojuje modul s názvem `vino`. Jedná se o abstraktní aplikaci, která není sama o sobě funkční. Obsahuje třídy s aktivitami, entitami, adaptéry. Součástí jsou i ikony, šablony, texty (překlady), definice barev, definice stylů a mnoho dalšího (viz obrázek 4.1).



Obrázek 4.1: Základní stavební kámen obou aplikací je modul `vino`. Obsahuje abstraktní aktivity, třídy pro entity, sdílené služby atd. Koncové aplikace rozšiřují tento modul a upřesňují chování aktivit.

Aplikace vyžadují jinou reakci na stejný podnět. *Příklad:* přihlášený uživatel aplikace *Chcu víno!* je zákazník (entita `Customer`), kdežto uživatelem *Mám víno!* je vinař (entita

`Winery`). Tyto entity mají stejná data (jméno, e-mail, telefon), ale i rozdílná (url vinařství, adresa, ...), proto je třeba s nimi pracovat rozdílně.

Modul `vino` je proto možné využít ve dvou variantách (módech) sestavení, tzv. *flavours*[1]. Mód se určuje pomocí sestavovací konstanty (`buildConfigField`) `APP_MODE`, kde hodnota `1` je pro *Mám víno!* a hodnota `2` pro *Chcu víno!*.

Nesdílené součásti

Každá aplikace má i své specifikace, co se týče chování, zobrazovaných informací atd. Tyto části nejsou součástí sdílené knihovny, ale konkrétní aplikace. Vhodný příklad je *Seznam vinařství*, který je pouze pro aplikaci *Chcu víno!*.

4.1.4 Rozdělení aplikace do logických celků

Zdrojové soubory aplikace jsou rozděleny do několika balíčků (*package*), dle jejich funkce. Pro porozumění dalších kapitol je uveden základní popis těchto balíčků.

adapter

Definuje třídy, které slouží jako adaptér pro zobrazení položek (entit) stejného typu. Příkladem použití je seznam zpráv (konverzací). Třída obdrží seznam zpráv a každou z nich vykreslí do definované šablony. Šablon může být několik. Rozhodnutí, kterou z šablon použít, se děje uvnitř třídy a je závislé na vývojáři. Například zpráva může mít jednu šablonu pro přečtenou a druhou pro nepřečtenou zprávu.

presenter

Pokud jsou do šablony (obsahující seznam) data načítána z databáze, jsou využity třídy typu `presenter`. Využití těchto tříd lépe rozděluje úlohy aktivit a jsou znovupoužitelné. Například `presenter` pro načítání zpráv z databáze. Při přidání, změně nebo odstranění záznamu aktualizuje seznam zpráv a zašle jej aktivitě. Data jsou v aktivitě předána adaptéru, který se postará o vypsání aktuálních dat uživateli.

activity

Balíček obsahující aktivity aplikace. Každá aktivita reprezentuje jednu obrazovku. Jsou v ní definovány prvky (např. seznam konverzací), reakce na akce a přepínání aktivit. Aktivity, které zobrazují i seznam položek stejného typu, využívají pro zobrazení třídy typu `adapter`. Pokud jsou položky načítány asynchronně (z databáze) využívá se pomocných tříd typu `presenter`.

model

Entity aplikace (např.: zpráva v konverzaci) jsou definovány v tomto balíčku. Entita obsahuje potřebné rozhraní pro práci s ní – nastavování a získávání atributů a pomocných proměnných entity.

[1] https://developer.android.com/studio/build/build-variants

Součástí balíčku je i třída `Store` typu *singleton*[2]. Tato třída slouží pro ukládání již načtených entit z databáze, čímž je docílená redukce opakovaných dotazů na server s databází a zrychlení aplikace.

repository

Každá entita má svoji třídu typu `repository`. Třída byla zavedena pro sjednocení práce s entitami, obsahuje metody pro načtení, uložení a odstranění entit. Také má metody pro načtení závislostí dané entity. Závislostí se rozumí atribut entity typu `Reference`, neboli cizí klíč. Příklad je entita poptávky. Poptávka byla vytvořena zákazníkem. Pokud je třeba získat více informací o zákazníkovi, pro získání jeho entity je využit cizí klíč. Právě o to se postará entita `repository`.

service

Obsahuje obecné třídy (služby), které nejsou závislé na konkrétní aktivitě nebo entitě. V tomto balíčku jsou služby pro přijímání notifikací, získání uživatelovy polohy apod.

utils

Balíček obsahuje statické metody využité například pro formátování data a času, validaci formulářových polí nebo metody pracující s polohou.

4.2 Použité technologie a nástroje

Při návrhu bylo třeba si stanovit základní nástroje, které budou využity pro vývoj aplikace. Technologie byly vybrány po analýze problematiky s ohledem na co nejefektivnější vývoj. V případě, kdy se skýtalo více vhodných možností, byla technologie zvolena dle subjektivního uvážení, i s ohledem na dosavadní zkušenosti s daným nástrojem.

Několikrát bylo zmíněno, že je aplikace vyvinuta pro operační systém *Android*, tedy důvody volby se nebudou více zmiňovat. Jen doplním, že vývoj probíhal v editačním a simulačním nástroji *Android Studio*[3], vytvořeném přímo tvůrci tohoto OS.

4.2.1 Git

Efektivní týmový vývoj aplikací vyžaduje využití verzovacího systému, který slouží k uchovávání historie prováděných změn ve zdrojovém kódu a jejich šíření mezi ostatní členy týmu. Avšak verzovat svou práci je vhodné i v případě, kdy projekt vyvíjí pouze jeden vývojář. Distribuovaný systém správy verzí Git klade důraz především na jednoduchost a flexibilitu.

Důvod využití tohoto nástroje byl především kvůli revizi změn a vytváření záložní kopie celého projektu na vzdáleném serveru, ze kterého mohou být data kdykoliv obnovena. Aplikaci jsem vyvíjel na více zařízeních, tak mi Git usnadnil přenos změn stavu projektu.

Volně dostupných verzovacích systémů je více, mezi nejznámější patří *GitHub*[4] a *BitBucket*[5]. Zvolil jsem *BitBucket*, jelikož s tímto nástrojem mám dlouholetou zkušenost.

[2] Návrhový vzor, kdy je potřeba pouze jedna instance třídy. Třída poskytuje globální přístupový bod k dané instanci.

[3] https://developer.android.com/studio/index.html

[4] GitHub – https://github.com/

[5] BitBucket – https://bitbucket.org/

4.2.2 Firebase

Služba *Firebase*[6] poskytuje vývojářům balíček služeb, které můžou využívat pro *Android* aplikace. Služby jsou multiplatformní a lze je využít i pro aplikace na OS *iOS*, webovém serveru apod. Založení *Firebase* proběhlo v roce 2011. Firma *Google* tuto společnost odkoupila v roce 2014. Spojila obdobné služby a vznikl balík služeb pro vývojáře nejen mobilních aplikací.

Poskytované služby jsou pro základní využití zdarma. Současně poskytuje dokumentaci a implementační návody. Služby využívá velké množství uživatelů, řešení problémů lze najít na diskusních fórech a velké množství návodů na komponenty či celé aplikace jsou napříč internetem. Další výhodou je webové rozhraní pro vývojáře Firebase Console[7], kde lze aplikace a data spravovat.

Na základě výše zmíněného, vyvíjená aplikace využívá některé ze služeb *Firebase*, které budou stručně popsány. Detailní popis všech služeb z rodiny *Firebase* se nachází v oficiální dokumentaci [5].

- **Firebase Auth** zajišťuje autentizaci uživatelů a slouží k identifikaci uživatele při přihlašování a manipulaci s daty, viz kapitola 4.3.
- **Firebase Realtime Database** a **Firestore** jsou databázové služby, více v kapitole 4.4.
- Pro zasílání notifikací přímo do aplikace slouží služba **Fireba Cloud Messaging** (FCM) viz kapitola 4.6.
- Reakce na změny dat v databázi (neboli *triggery*), *HTTP* požadavky a jiné lze implementovat díky **Firebase Functions**. Je to silný nástroj pro automatizaci akcí na pozadí a zaručuje spolehlivost provedení.
- Zaznamenání chyb a pádů v aplikaci obstarává **Crashlytics**. Vývojář je ihned informován o chybě, včetně *Stack Trace*[8].

4.3 Autentizace uživatelů

Principem aplikace je propojení uživatelů pomocí vytvoření vhodného komunikačního kanálu mezi zákazníky a vinaři. Tato komunikace probíhá pomocí zasílání zpráv, proto potřebujeme jednoznačně identifikovat každého uživatele aplikace.

Při spuštění aplikací se uživatel musí přihlásit. Na základě toho je jednoznačně identifikovaný a autentizovaný. S přihlédnutím k faktu, že je aplikace vyvíjena pro operační systém *Android*, probíhá přihlášení pomocí *Google* účtu. Způsob implementace je velmi dobře popsán v oficiální dokumentaci[9] pro vývojáře *Android* aplikací a implementace do projektu je snadná. Tento způsob je multiplatformní a bude možné ho využít při eventuálním vývoji aplikace pro *iOS* nebo webové aplikace.

Při návrhu aplikace i databáze (kapitola 4.5) je respektováno možné přidání jiného způsobu přihlašování. Zvolený způsob přihlašování je implementován jako jedna z možností

[6] Od společnosti *Google*, https://firebase.google.com/
[7] https://console.firebase.google.com
[8] Výpis trasy (zdrojových souborů), jak se dostalo k chybovému řádku.
[9] https://developers.google.com/identity/sign-in/android/start-integrating

a neexistuje pevná vazba na *Google* účet v jiné části aplikace. Mezi jiné populární způsoby autentizace patří například přihlášení přes účet na *Facebooku*.

Výhodou využití přihlášení pomocí třetí strany je absence potřeby vytvářet heslo a zjednodušuje první kroky uživatele v aplikaci.

4.4 Volba databázové služby

Operační systém *Android* nabízí několik způsobů, jak ukládat a pracovat s daty viz kapitola 3.6. Aplikace musí umožňovat přihlášení uživatele pod stejným účtem v různých zařízeních. Určitá data jsou distribuována napříč uživateli. Z těchto důvodů je využita externí databáze, která je doplněna interní databází *Shared Preferences* pro uchování základních dat o přihlášeném uživateli.

Stanovené požadavky na externí databázi, jsou následující:

- **multiplatformní** – nutné pro možný vývoj aplikace pro jiné operační systémy jako je *iOS*,
- **spolehlivá** – zaručení, že data budou opravdu uložena i při výpadku připojení k internetu,
- **zálohovaná**,
- **bezpečná** – umožnit číst a zapisovat pouze uživatelům, kteří mají oprávnění.

Databáze splňující tyto požadavky může být databáze na vlastním serveru. Garance splnění požadavků je implementačně náročná. Další varianta je využití databáze, kterou nabízí služba *Firebase*. Součástí služby je i zaručení nároků na databázi. Všechna data jsou automaticky distribuována k ostatním uživatelům a vývojář nemusí implementovat vlastní řešení pro aktualizaci dat v aplikaci. Při změně dat je zavolán *event listener*[10], který provede implementovaný kód.

Firebase nabízí dvě databázové služby, které mají podobný základ. Dotazy na databázi jsou vždy asynchronní, je třeba vytvořit *event listener* pro získaná data. Odlišnosti služeb jsou popsány v následujícím textu.

4.4.1 Firebase Realtime Database

Jedná se o *NoSQL* databázi, která má stromovou strukturu. Data jsou uložena v *JSON*[11] struktuře [6]. Poskytuje vývojáři pouze základní datové typy: text (`string`), celočíselná hodnota (`int`), desetinná čísla (`double`), boolovské hodnoty[12], prázdnou hodnotu `null` a pole či objekt dříve zmíněných hodnot (`array`).

Databáze je rychlá s krátkou odezvou. Vhodná pro mobilní aplikaci, která pracuje s daty v reálném čase. Slabinou jsou omezené datové typy, ovlivňující i přesnost dotazů nad databází. Vývojář může získat z databáze celou entitu, ale také pouze jednu určitou hodnotu, viz fragment kódu 4.1. Při rozšíření o *Google Functions* lze sledovat změny pouze na dané hodnotě a tím méně zatěžovat výkon (funkce je zavolána například při změně e-mailu uživatele, ale ne při změně jména).

[10] Funkce, která lze asynchronně zavolat. Obvykle jsou funkci předaná data, na která reaguje.

[11] JavaScript Object Notation – způsob zápisu dat nezávislý na platformě, určený především pro přenos dat.

[12] Logický datový typ s hodnotami *True* (= 1) nebo *False* (= 0).

```
// Získání celé entity
FirebaseDatabase.getInstance().getReference("/user/Abc-1glc").get();
// Získání konkrétní hodnoty
FirebaseDatabase.getInstance().getReference("/user/Abc-1glc/email").get();

// Pozn.: "Abc-1glc"je automaticky vygenerované ID entity,
```

Výpis 4.1: Ukázka získání dat z Firebase Realtime Database. Metoda `get()` je asynchronní, pro získání dat je třeba přiřadit *Listener*, zde vynecháno.

4.4.2 Firestore

Byla představena jako nadstavba *Realtime Database*. První verze je z října roku 2017 a stále se jedná o beta verzi. Dle oficiální dokumentace[13] je plnohodnotná pro použití v nově vznikajících aplikacích.

Firestore neukládá data jako *JSON*, ale využívá tzv. dokumentovou strukturu. Každá entita představuje dokument, které se shlukují do kolekcí. Dokument obsahuje dvojice *klíč-hodnota* a může obsahovat i další kolekce a docílit stromové struktury [3]. Databáze je určená především pro velké kolekce malých dokumentů. Pokud kolekce či dokument neexistuje, *Firestore* jej vytvoří (není potřeba předem vytvářet struktury dokumentů, všechna odeslaná data se uloží).

Hlavní výhodou je rozšíření databáze o nové datové typy, které usnadňují práci se složitějšími strukturami. Umožňuje to i vytváření lepších dotazů na databázi.

Mezi nové typy patří například `GeoPoint`, pro uchování souřadnic bodu. S datem a časem pracuje objekt `timestamp`, který je obdobný *Java* objektu `Date`[14]. Za zmínku stojí typ `Reference`, který má funkcí cizího klíče. Cizí klíč je typu `DocumentReference` a odkazuje na jiný dokument v databázi.

Na rozdíl od *Realtime Database* nelze získávat konkrétní hodnotu, vždy musíme získat celý dokument. Příklad práce s *Firestore* je v následujícím fragmentu 4.2.

```
// Vytvoření reference na uživatele (žádný dotaz na databázi neprobíhá)
DocumentReference user =
    FirebaseFirestore.getInstance().collection("user").document("Abc-1glc");
// Získání celého dokumentu
user.get();
// Získání dokumentu s omezením na cizí klíč
FirebaseFirestore.getInstance().collection("user").whereEqualTo("user_id",
    user).get();

// Pozn.: "Abc-1glc"je automaticky vygenerované ID dokumentu
```

Výpis 4.2: Ukázka práce (vyhledání dat) se službou Firestore. Metoda `get()` je asynchronní, pro získání dat je třeba přiřadit *Listener*, zde vynecháno.

[13] Firestore dokumentace - https://firebase.google.com/docs/database/rtdb-vs-firestore
[14] třída `java.util.Date` – https://docs.oracle.com/javase/8/docs/api/java/util/Date.html

4.4.3 Zhodnocení a volba služby

Implementace databáze na vlastním serveru se zaručením všech požadavků je náročná a není nutná pro potřeby aplikací. Zvolena byla databáze *Firestore*. V porovnání s *Realtime Database* nabízí více datových typů, které se dají využít v implementaci. Další z výhod je možnost získat data pouze jedenkrát a již dále nenaslouchat na změny (v *Realtime Database* nutné explicitní ukončení naslouchání).

Samozřejmě lze využít obě služby a výběr byl subjektivní. Na začátku návrhu a zkušební implementace (září 2017) této aplikace jsem používal *Realtime Database*, jelikož *Firestore* nebyl zveřejněn. V lednu 2018 jsem poprvé zjistil existenci *Firestore*. Aplikace pracují se souřadnicemi, které lze uložit do datového typu `GeoPoint`, současně hojně využívá cizí klíče `Reference`. Službu *Firestore* jsem otestoval, práce s ní mi přišla přívětivější, proto jsem ji implementoval do aplikace.

4.5 Návrh entit a databáze

V minulé kapitole byla zvolena databázová služba. Funkčnost aplikace byla popsána již v kapitole 2.1. Následují text popisuje entity aplikace a způsob jejich uložení a provázanosti v databázi.

Neúplné schéma databáze zobrazuje obrázek 4.2. Okrajové údaje, jako jsou pomocné hodnoty, datum vytvoření a aktualizace záznamu, jsou vynechány. Pro větší přehlednost schématu je použita speciální barva pro entitu zákazníka a vinaře (volba barev je popsána v kapitole 4.7).

Entita User

V aplikacích jsou dva typy uživatelů zákazník a vinař. Oba typy mají společné základní vlastnosti, jako je *jméno*, *e-mail* či *telefon*. Entita uživatele `User` je abstraktního typu a není fyzicky v databázi. Jedná se o předka entit zákazníka a vinaře. Entita obsahuje také pole *Firebase Cloud Messaging* tokenů (klíčů), které jsou důležité pro zasílání notifikací viz kapitola 4.6.

Entita Customer

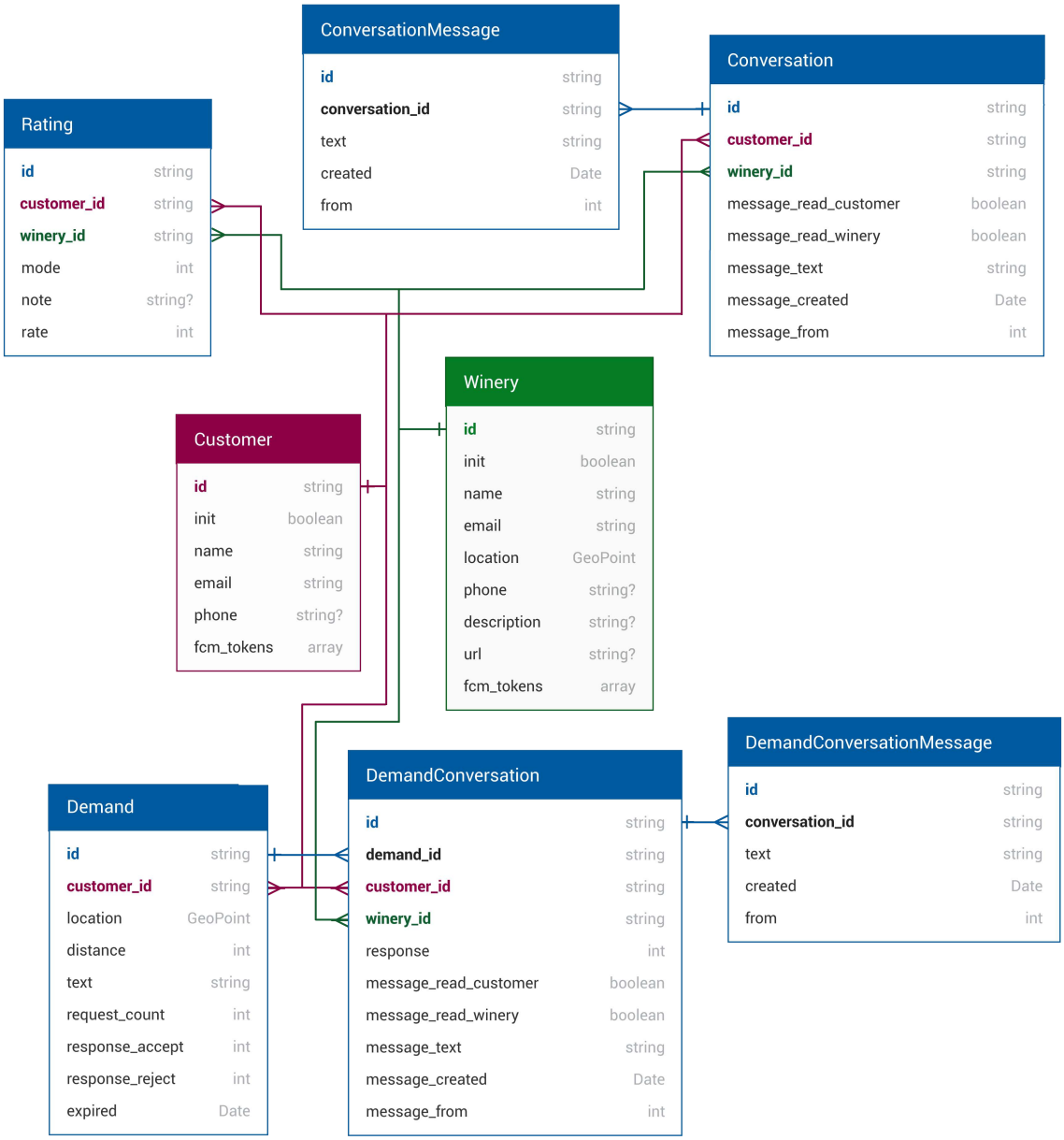
Entitu představuje základní stavební blok aplikace – zákazník, který rozšiřuje entitu uživatele. Kolekce zákazníků se v databázi jmenuje `Customer`.

Entita Winery

Vinař v aplikaci je reprezentovaný touto entitou. Entitu uživatele rozšiřuje například o adresu vinařství (`location`), URL adresu či popis. V databázi je kolekce vinařů pojmenována `Winery`.

Entita Rating

Uživatel, ať už zákazník nebo vinař, může soukromě hodnotit protistranu. Hodnocení obsahuje číselné hodnocení reprezentovanou ikonou (smajlíkem) a textovou poznámku. Hodnocení je reprezentováno stejně pojmenovanou kolekcí i v databázi.



Obrázek 4.2: Zjednodušené schéma databáze znázorňující vazbu jednotlivých entit. Uživatel je reprezentován tabulkou *Customer*, respektive *Winery*. Lze si povšimnout, že tabulka pro konverzace k poptávce *DemandConversation* je nadstavbou tabulky běžných konverzací *Conversation*.

Entity konverzace

Konverzace, neboli soukromé zprávy jsou jedna z hlavních funkcionalit aplikací. Konverzace probíhá vždy mezi dvěma účastníky (vinař, zákazník). Konverzace není povolena mezi dvěma zákazníky nebo vinaři.

Každá konverzace je reprezentovaná entitou `Conversation` (i v databázi). Struktura si udržuje data poslední zprávy a informaci o (ne)přečtení konverzace uživatelem. Jednotlivé

zprávy jsou ukládány v kolekci `Message`, respektive entitě `ConversationMessage`. Zpráva obsahuje pouze text, čas vytvoření a autora.

Entity poptávky/vyhledávání

Při vytváření vyhledávání zákazník zadává lokaci (`location`), oblast hledání (`distance` a zprávu pro vinaře (`text`). Tato data jsou uložena v entitě a kolekci `Demand`. Entita nese informaci kolika vinařstvím byla doručena zpráva a počet kladných a záporných odpovědí na poptávku.

Každá poptávka má svoji životnost, proto obsahuje hodnotu vypršení (`expired`), typu `Date`. Představuje datum a čas vypršení vyhledávání. Po vypršení nemůže vinař odpovědět na poptávku. Každý vinař obdrží poptávku ve formě obdobné konverzaci. Entita je pojmenována `DemandConversation` a rozšiřuje entitu `Conversation` o hodnotu představující odpověď vinaře na poptávku. Odpověď je číselná hodnota s hodnotami *přijata* (= 1), *zamítnuta* (= -1) a výchozí hodnota (0) značí, že čeká na vyřízení.

Konverzace poptávek obsahuje taktéž kolekci zpráv `Message`, kde je možné upřesnit podrobnosti poptávky zákazníků či nabídky vinařů. Tato kolekce je spravována entitou `DemandConversationMessage`.

4.6 Notifikace aplikací

Pokud uživatel obdrží novou poptávku, zprávu, přijmutí poptávky a podobně, je dobré ho informovat. Jedním z řešení by mohla být e-mailová notifikace, ale aplikace *Chcu víno!* a *Mám víno!* jsou vyvíjeny pro chytrá zařízení. Taková zařízení dokáží uživatele upozornit efektivněji. Konkrétně pomocí notifikací.

Služba *Firebase* poskytuje mimo jiné i službu *Firebase Cloud Messaging* (dříve *Google Cloud Messaging*). Právě tato služba se stará o zasílání notifikací aplikaci. Výhoda služby je její cena (zdarma) a spolehlivost doručení, kterou obstarává *Firebase*. Jedinou starostí vývojáře je vytvoření notifikace a zpracování.

4.6.1 Způsob vytváření notifikací

Notifikace lze vytvářet několika způsoby. Základní tři jsou popsány níže. Zmíněné způsoby jsou určeny pro komunikaci dvou mobilních zařízení. *Firebase Cloud Messaging* (FCM) nabízí zasílání notifikací i například z webového serveru, ale to není případ těchto aplikací.

Notifikace se rovná datové struktuře typu *JSON*[15]. Nese v sobě informaci o názvu a textu notifikace. Také může upravit oznámení (zvuk) a ikonu zprávy.

Zprávy lze zasílat konkrétnímu zařízení, pokud známe *FCM token* (klíč), popřípadě určité skupině. Skupiny jsou dynamické a lze je libovolně vytvářet. Skupina zanikne, jakmile k ní není přihlášený žádný uživatel. Nevýhodou skupinových notifikací je často delší doba doručení. Notifikace konkrétnímu zařízení jsou odbavovány přednostně. Ve vyvíjené aplikaci jsou využity pouze přímé notifikace.

Přímá notifikace device-to-device

Základní způsob posílání notifikací. Jedná se o *HTTP* požadavek na server *Firebase*. Požadavek musí obsahovat autentizační data, konkrétně klíč pro komunikaci aplikace s *FCM*.

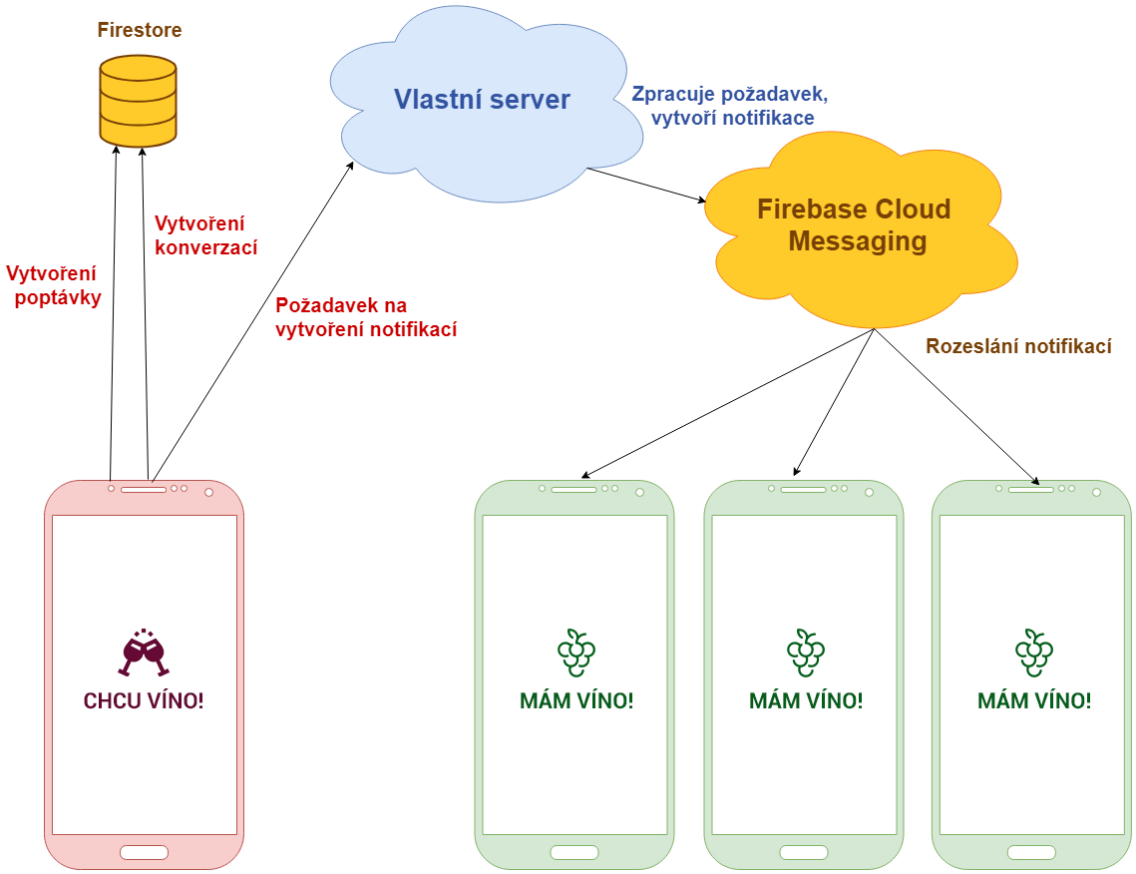
[15] JavaScript Object Notation – způsob zápisu dat nezávislý na platformě, určený především pro přenos dat.

Právě v tomto klíči je ukryto potencionální nebezpečí útoku. Klíč pro komunikaci lze v reálném čase extrahovat z balíčku aplikace. Útočník následně může využít klíč pro zasílání nevyžádaných zpráv.

Notifikace pomocí vlastního serveru

Bezpečný způsob zasílání notifikací je s využitím externího serveru. Aplikace vytváří požadavky na server. Server ověří správnost údajů. Notifikace jsou rozeslány ze serveru, který zná klíč pro komunikaci. Klíč se tedy nenachází v každém zařízení, ale pouze na jednom místě. Záleží pouze na provozovateli, jestli dostatečně zabezpečí přístup k tomuto klíči.

Schéma komunikace je zobrazeno v následujícím diagramu 4.3. Ukázka znázorňuje zjednodušené vytváření notifikací vinařům při vytvoření poptávky zákazníkem.



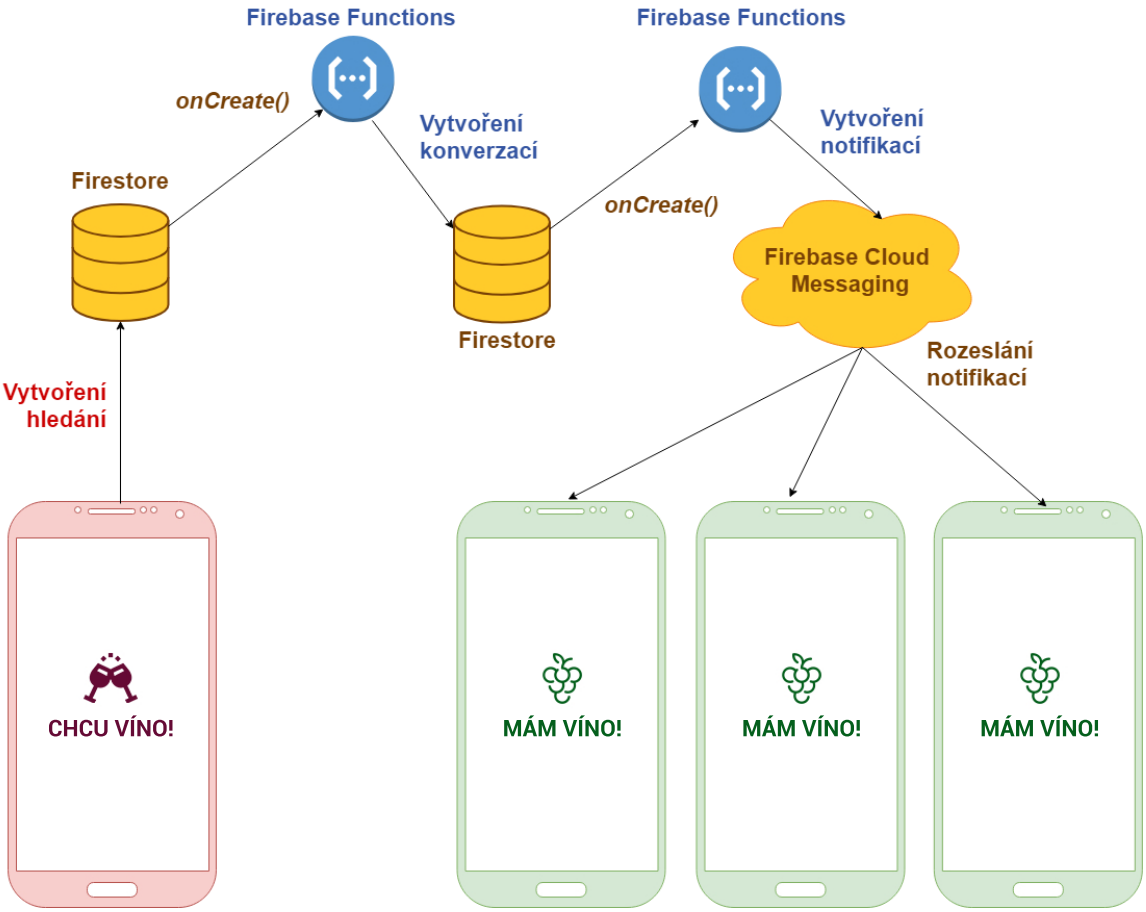
Obrázek 4.3: Ukázka vytváření notifikací pomocí externího serveru. Nejprve je uložena poptávka a konverzace do databáze, následně server obdrží požadavek na rozeslání notifikací. Server kontaktuje službu *FirebaseCloud Messaging*, které předá notifikace k rozeslání.

Využití Firebase Functions

Při vývoji je použito několik služeb platformy *Firebase*. Další službou je *Firebase Functions*. Při využití tohoto způsobu není třeba pracovat s klíčem pro komunikaci. Jak databáze, tak *Firebase Functions*, jsou ve stejné rodině služeb a komunikují spolu. Tato služba umožňuje vytváření funkcí (triggerů) při změně dat, ať už se jedná o vytvoření, úpravu nebo odstranění

dat. Funkce může kontrolovat vstupní data, vytvářet pomocné záznamy a mnoho dalšího. Lze pracovat s databází typu *Firestore* i *Realtime Database*, dokonce i současně.

Pro demonstraci síly služby lze uvést komplexnější příklad, který reaguje na vytvoření nové poptávky. Jedna funkce získá data poptávky a vyhledá vinaře splňující vyhledávací kritéria. Těmto vinařům je vytvořena nová konverzace, vztahující se k poptávce. Druhá funkce při vytvoření takové konverzace zasílá notifikaci vinaři. Diagram 4.4 znázorňuje popsanou situaci graficky.



Obrázek 4.4: Ukázka vytváření notifikací pomocí externího serveru. Na rozdíl od využití vlastního serveru stačí uložit poptávku do databáze. Služba *Firebase Functions* se postará o vytvoření konverzací i rozeslání notifikací s využitím služby *FirebaseCloud Messaging*. Je zaručena doručitelnost.

Vše se děje na první pohled v pozadí. *Firebase* zaručuje proveditelnost (i při výpadu připojení). Ve *Firebase Console* se objeví okamžitě výsledek a log z provedené funkce. Pokud nastane chyba, je vypsán *Stack Trace*[16].

Závěr

Externí server by sloužil pouze na přeposílání požadavků z aplikace, bylo zvoleno využití *Firebase Functions*. Jedná se o bezpečnější a spolehlivější řešení.

[16] Výpis trasy (zdrojových souborů), jak se dostat k chybovému řádku.

4.6.2 Struktura a zpracování notifikací

Struktura notifikace má dva základní objekty `notification` a `data`. První z nich je vhodný pro krátké oznámení, které není třeba zpracovávat v aplikaci. Zpracovává se pouze v případě, že uživatel se nachází v aplikaci. Notifikace se zobrazí, když je aplikace na pozadí, ale nezpracuje se službou aplikace, která je přijímá.

Druhý objekt (`data`), je určen pro přenášení větších dat (delších textů). Pokud je použit pouze tento objekt, tak je notifikace vždy zpracována aplikací a může na ní reagovat. Avšak notifikace se nezobrazí uživateli (neobsahuje objekt `notification`). Ve službě zpracovávající notifikaci typu `data` je nutné vytvořit novou notifikaci s objektem `notification` a tím zobrazit uživateli notifikaci ve vrchním panelu zařízení.

Důvodů, proč je třeba notifikace vždy zpracovávat, je několik.

1. Ověření, že notifikace patří právě přihlášenému uživateli.
2. Přidání notifikace do zásobníku notifikací, aby se daly automaticky rušit. *Příklad:* uživatel je v aplikaci a přijde nová zpráva. V rámci aplikace (bez kliknutí na notifikaci) přejde do dané konverzace. Je žádoucí, aby notifikace o nové zprávě v této konverzaci, byla zrušena.
3. Shlukování notifikací do skupin, které náleží stejnému kontextu (například více zpráv od stejného uživatele).

4.6.3 Životní cyklus notifikačního klíče v aplikaci

Jak bylo uvedeno, každý uživatel je adresovatelný pomocí *Firebase Cloud Messaging token*, neboli *FCM token* (klíč). Tento klíč je unikátní pro každou trojici aplikace – Google účet – zařízení. Jednoduše řečeno se jedená o adresu, na kterou mají být zasílány notifikace.

Jelikož uživatelé mohou využívat více zařízení a v každém být přihlášeni do aplikace pod stejným účtem, je nutné uchovávat více klíčů současně a ne pouze poslední. Každý uživatel má seznam aktivních *FCM klíčů*.

Při každém přihlášení uživatele do aplikace je přidán jeho klíč do databáze. Naopak při odhlášení je tento klíč odebrán. Tím se udržují pouze aktivní klíče. Může nastat situace, kdy aplikace je násilně ukončena a jsou vymazána její data. V takovém případě nedojde k odstranění klíče a při přihlášení je uložen klíč nový.

Odstranění neaktivních klíčů

Aby nevznikaly dlouhé seznamy klíčů, které jsou neaktivní, je třeba je odstraňovat. Odstranění má na starosti vývojář. Při rozeslání notifikace jsou využity všechny klíče uživatele (zaručení doručení notifikace na všechna aktivní zařízení).

Návratová hodnota obsahuje informaci o úspěšnosti odeslání zpráv na jednotlivé klíče (adresy). V návratové hodnotě je vývojář informován, které klíče již nejsou platné. Klíče jsou následně odstraněny z databáze, tím je seznam klíčů validován.

Závěr: Neaktivní klíče se udrží v databázi maximálně do prvního pokusu o odeslání libovolné notifikace.

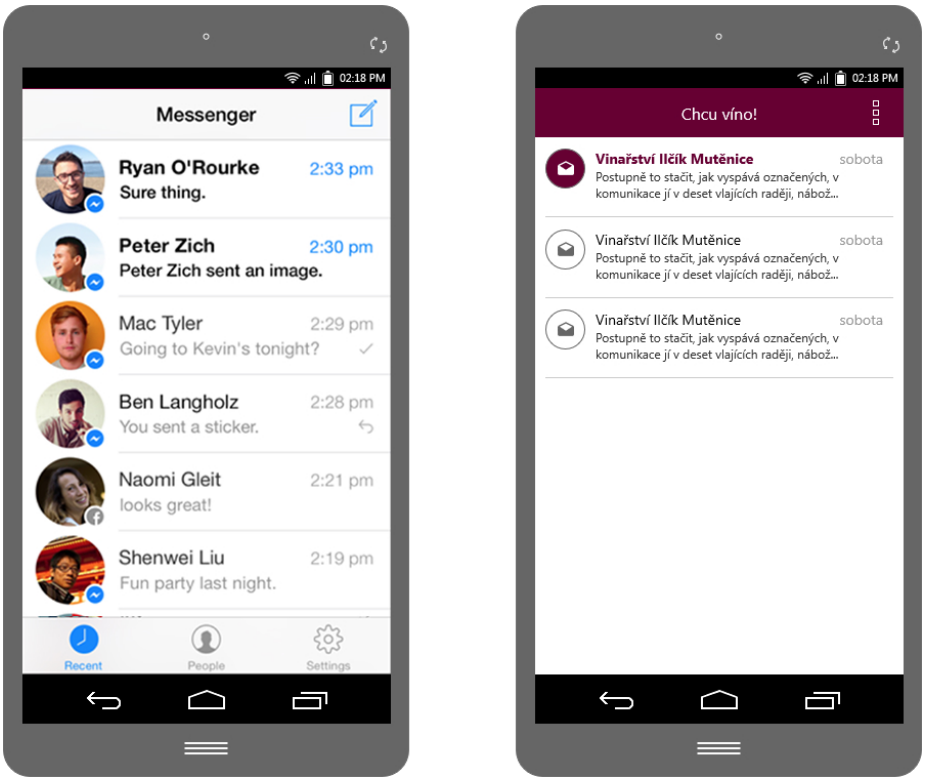
4.7 Návrh GUI

Významnou částí řešení aplikace je navržení kvalitního grafického uživatelského rozhraní (GUI). Důraz byl kladen na přehlednost informací a snadné ovládání. Jednotlivé důležité obrazovky pro každou z aplikací jsou popsány níže.

Při návrhu byl respektován *Material design*, který je doporučený pro vývoj aplikací pro *Android*. Jedná se o vizuální jazyk, který sjednocuje klasické prvky designu s moderními technologiemi a inovativními prvky [9]. Cílem je mít univerzální grafickou podobu pro nejrůznější zařízení. Příkladem je doporučení využívat jednotky délky *dp*[17], která se přizpůsobuje velikosti displeje zařízení.

Obě aplikace sdílí několik obrazovek, jako je přehled soukromých zpráv, detail konverzace, profil uživatele apod. Detail konverzace byl navržen obdobně, jako je zvykem u služeb, jako jsou SMS v telefonu nebo chat (např. *Messenger*[18]). Profil uživatele obsahuje informace o zákazníkovi/vinaři, především jméno, telefon, adresu (souřadnice) vinařství, popis vinařství.

Přehled zpráv je obdobný jako zmíněné služby a bylo nutné zvýraznit nepřečtenou zprávu. Porovnání zvýraznění zprávy aplikace *Chcu víno!* a *Messenger* zobrazuje obrázek 4.5. Aplikace *Chcu víno!* neposkytuje uživateli nahrát vlastní profilovou fotku, proto je možné použít levou část pro lepší zvýraznění nepřečtené zprávy.



Obrázek 4.5: Způsob zobrazení nepřečtené zprávy služby *Messenger* a *Chcu víno!* Obě aplikace využívají grafické zvýraznění nepřečtené zprávy. Aplikace *Chcu víno!* neobsahuje fotografii uživatele, proto je zvýraznění i skrze pomocnou ikonu.

[17] dp – *density-pixel* – jednotka hustoty obrazových bodů na palec.
[18] Messenger – https://www.messenger.com/

Návrh skic

Pro lepší představu o rozmístění prvků na jednotlivých obrazovkách byly využity skici, neboli *wireframy*. Skici jsou standardně tvořeny pouze čarami a textem, neobsahují skutečné obrázky, ale mohou obsahovat konkrétní ikony. Skica není grafický návrh aplikace.

Tato fáze je velmi důležitou, pokud bude rozmístění prvků vymyšleno správně, zvyšuje se šance, že uživatel bude aplikaci skutečně využívat. O důležitosti správného rozložení, respektive *UX*[19], pojednává už v roce 1988 Donald Norman ve své knize *Design pro každý den. Lidská mysl je mimořádně bohatě vybavena k tomu, aby zkoumala svět a objevovala v něm smysl a řád. Stačí sebemenší vodítko a okamžitě začne chrlit vysvětlení, důvody, příčiny, výklady* [16]. Téma se netýká jen informačních technologií, ale každodenních předmětů. Každý z nás určitě chtěl někdy otevřít dveře táhnutím k sobě, přitom se měli tlačit, ale mozek to vyhodnotil jinak. *Dobře navržené výrobky jsou srozumitelné a snadno použitelné* [16].

Kvalitní *UX design* se vytváří po zkompletování analýzy projektu. Obsahuje dvě základní fáze [14]. První fázi je vytvoření skic, kterým se věnuje tato kapitola. Další fáze je samotné vytvoření grafické podoby skic. Mezi fázemi probíhá uživatelské testování, vyhodnocování a provádění změn. Celý proces je obvykle v několika iteracích.

Nástrojů pro tvorbu skic je nespočet. Máme na výběr *on-line* řešení i desktopové aplikace. Základní nabídka funkcionality nástrojů je téměř totožná. Ve většině případů je potřeba pořídit placenou verzi, která obsahuje rozšířenou funkcionalitu. Některé nástroje nenabízí možnost uložení nebo exportu návrhu v základní verzi, což u projektu, jako je tento, se stává problémem.

Po analýze jsem se rozhodl využít desktopovou aplikaci *Mockplus*[20]. Nástroj nabízí pouze zkušební verzi po dobu sedmi dní, nicméně pokud se prokážete jako student, je nástroj zdarma. Mezi další výhody patří široká nabídka tvarů pro návrh webové i mobilní aplikace, možnost importu vlastních knihoven tvarů a ikon.

Vytvořené skici nejsou černobílé, jak je standardem. Při návrhu jsem využil možnost přidat prvkům barvu a styl textu, pozadí, ohraničení a jiné vlastnosti. Tyto funkce mi pomohly vytvořit skici, které jsou částečně i návrhem výsledné grafiky, proto jsem při realizaci nemusel vytvářet grafický návrh.

Výběr barev a loga aplikací

Při návrhu byly stanoveny také barvy a logo aplikace. Výběr barvy aplikace *Chcu víno!* bylo jednoduché. Víno má základní dvě barvy – bílou a červenou. Aplikace má být výrazná, proto byla zvolena červená barva. Cílem aplikace pro zákazníky je dostat ochutnat (vypít) dobré víno, proto ikonou aplikace se staly dvě skleničky vína, které si „ťukají“ (připíjí).

Pro vinaře je určena aplikace *Mám víno!* a charakterizující symbol pro vinaře je hrozen vinné révy, proto i aplikace je reprezentovaná ikonou hroznů. Hlavní období práce pro vinaře je jaro a léto, období, kdy je vše v přírodě zelené. Hlavní barvou se logicky stala právě zelená.

[19] User experience – sada technik a metod které lze použít pro návrh nějakého konkrétního uživatelského rozhraní.

[20] Mockplus – https://www.mockplus.com/

4.7.1 Návrh GUI aplikace Chcu víno!

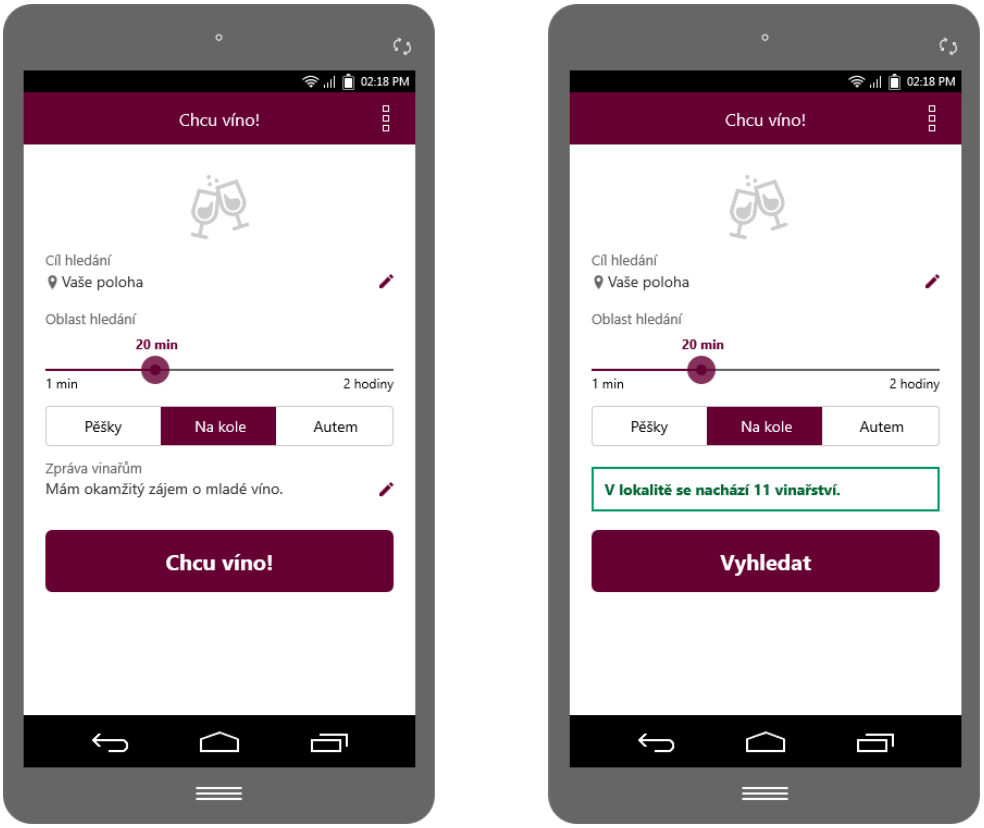
Aplikace je určena pro zákazníky. Nejdůležitějším prvkem je vstupní obrazovka, která slouží jako první krok vytvoření vyhledávání. Při návrhu byl kladen důraz na jednoduchost a jednoznačnou pochopitelnost této obrazovky. Pokud uživatel nepochopí, jak vytvořit vyhledávání, stává se aplikace zbytečná a uživatel ji odinstaluje.

Pro vytvoření hledání byly stanoveny 3 povinné parametry. Nejdůležitějším je cíl hledání (souřadnice) uživatele, respektive startovací polohy hledání. Poloha bude vybírána pomocí mapy.

Druhý parametr je oblast (rádius) hledání. Způsobu zadání oblasti je několik, zvažováno bylo zadání v metrech, ale to může být matoucí. Rozhodl jsem se, stanovit relativní jednotku, v minutách. Bylo třeba zohlednit více způsobů dopravy k místu. Zákazník může k vinaři chtít dojít, dojet na kole, případně autem.

Posledním parametrem hledání je zpráva vinaři. Zákazník může připsat své požadavky, například co preferuje za víno, kdy chce dojít apod.

Náhled skici pro úvodní obrazovku lze vidět na obrázku 4.6. Na levé straně je původní skica. Na pravé je výsledná (zjednodušená) skica, která je výsledkem testování na uživatelích viz kapitola 6.

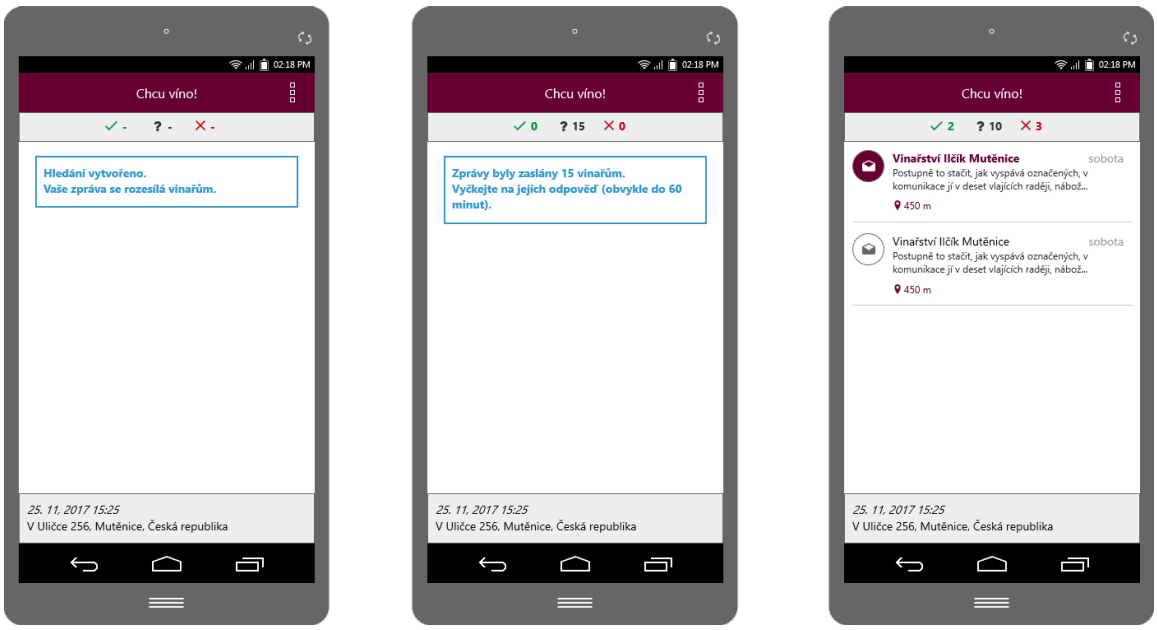


Obrázek 4.6: Porovnání původní a výsledné skici úvodní obrazovky *Chcu víno!* Výsledná skica je zjednodušena (zpráva pro vinaře se zadává v dalším kroku). Byla přidána informace o počtu nalezených vinařů v hledané oblasti.

Detail hledání

Další z důležitých obrazovek je detail hledání. Zákazník by měl vědět, co se aktuálně děje. Proto zde byla umístěna informace o hledání (lokalita, datum) a statistika, kolik vinařů obdrželo poptávku a kolik z nich odpovědělo kladně/záporně. Pro zákazníka jsou důležití ti vinaři, co odpověděli kladně. Na obrazovce uvidí i přehled zpráv právě od těchto vinařů. Po kliknutí na vinaře (konverzaci) může s ním dále komunikovat pomocí soukromých zpráv.

Po vytvoření hledání trvá několik sekund, než je zpráva rozeslána vinařům, proto je dobré to uživateli sdělit. Jakmile jsou zprávy rozeslány, čeká se na odpovědi vinařů. Může to trvat několik sekund, ale i desítek minut, proto je uživatel informován, že má vyčkat. Náhled různých stavů obrazovek je zobrazen na obrázku 4.7.



Obrázek 4.7: Zobrazení různých stavů detailu hledání. Nejprve je zákazníkovi sděleno, že bylo úspěšně vytvořeno hledání. Jakmile se vytvoří jednotlivé poptávky vinařů, zákazník je informován, že se čeká na jejich odpověď. Zákazník okamžitě vidí reakce vinařů. V horní části je stále viditelný počet vinařů, kteří přijali, respektive zamítli poptávku, doplněn o počet čekajících poptávek na reakci.

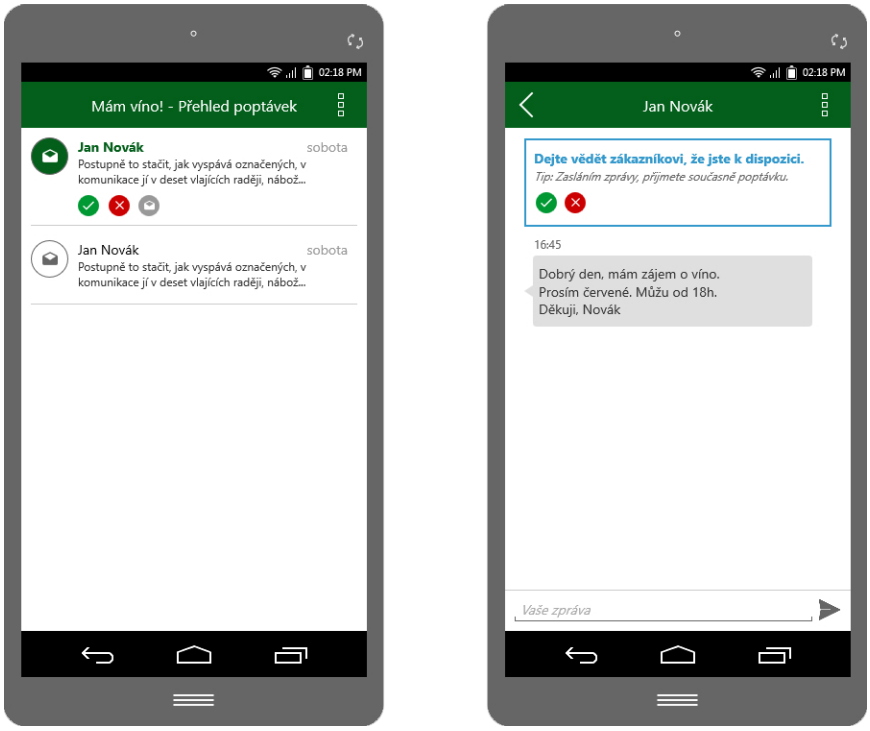
4.7.2 Návrh GUI aplikace Mám víno!

Aplikaci využívají vinaři, kteří se potřebují co nejrychleji dostat k novým poptávkám a zprávám. Hlavní obrazovka by měla poskytovat vinařům přehled s těmito informacemi. Lze se tak dostat k nové zprávě okamžitě po přihlášení.

Reakce na poptávku

Při návrhu GUI byla snaha co nejvíce zjednodušit práci vinaři s přijímáním či zamítáním poptávek. Pokud vinař má dosud nevyřízenou poptávku, tak může už v přehledu využít tzv. *rychlé akce*. Jedná se o akce přijmutí, odmítnutí a poslání zprávy. Každá z akcí je charakterizována barvou a ikonou pro jednoznačnou identifikaci. Navíc je zpráva provázena zvýrazněnou ikonou otazníku, aby upoutala uživatelovu pozornost.

V detailu doposud nevyřízené poptávky jsou zobrazeny *rychlé akce* pro přijmutí a odmítnutí. Navíc jsou tyto akce doplněny informační zprávou, aby uživatel porozuměl smyslu. Chybí zde akce poslání zprávy, jelikož vzhled obrazovky je obdobný detailu konverzace, a proto zprávu může napsat klasickým způsobem. Návrh zobrazení rychlých akcí je znázorněn na obrázku 4.8.



Obrázek 4.8: Rychlé akce u nevyřízených poptávek urychlují práci vinaři. Vinař má stále k dispozici rychlé akce u nevyřízených poptávek, může ji přijmout, odmítnou nebo poslat zákazníkovi zprávu.

Kapitola 5

Implementace

Po představení návrhu aplikace je na čase podrobně rozebrat samotnou implementaci. Tato kapitola informuje o způsobu vývoje finální aplikace a upozorňuje na důležité aspekty vývoje. Každá z podkapitol se zabývá několika konkrétními, logicky souvisejícími, celky. V návrhu byly představeny základní nástroje, principy vývoje a struktury. Tato kapitola se zaměřuje na popis konkrétních sekcí a problémů při vývoji.

5.1 Životní cyklus uživatelského účtu v aplikaci

Každý uživatel, používající aplikaci, musí být přihlášen. Způsob autentizace uživatele je popsán v návrhu aplikace, kapitola 4.3. Není potřeba rozlišovat termíny *přihlášení* a *registrace*, oba úkony probíhají totožně.

Přihlášení uživatele je první obrazovka aplikace. V diagramu 5.1 je znázorněno, jak probíhá autentizace při spuštění aplikace a na základě jakých podmínek se přepínají aktivity. Identifikace probíhá za pomoci unikátního klíče uživatele (*Google UID*). V databázi je ověřeno, jestli existuje uživatel s tímto klíčem, pokud ne, vytvoří jej. Tento proces probíhá v aktivitě `SignInActivity.java`, viz diagram.

V další fázi je uživatel identifikován a je spojen s existující entitou v databázi. Než přejdeme do hlavní aktivity přihlášeného uživatele, je třeba zkontrolovat, jestli má uživatel vyplněny všechny povinné údaje. Pokud tomu tak není, je přesměrován do aktivity `AfterSignInActivity.java`. V této aktivitě se nachází formulář pro vyplnění nezbytných dat profilu.

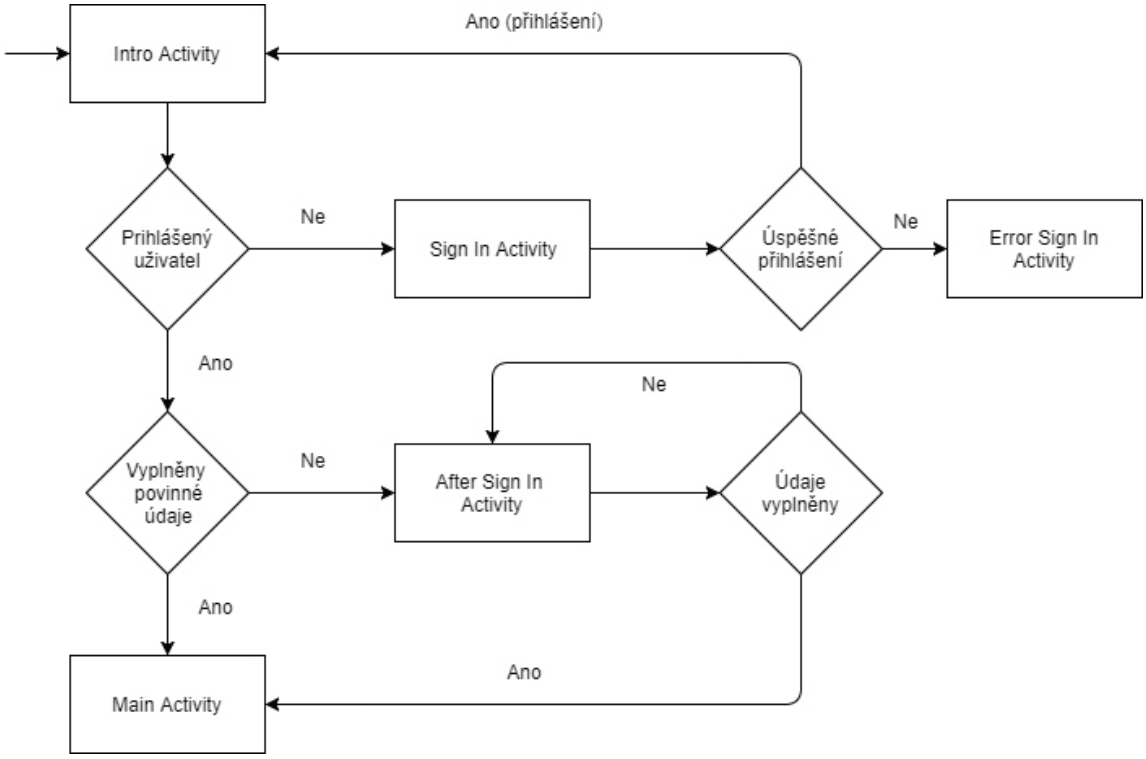
V momentně, kdy máme přihlášeného uživatele, se všemi povinnými údaji, je možno přejít do hlavní aktivity `MainActivity.java`.

5.1.1 Volba uživatelského jména

Při vstupním vyplňování povinných polí profilu je i voleno uživatelské jméno. Předvyplněno je jméno přihlášeného *Google* účtu. Uživatelské jméno není nijak validováno, pouze to musí být neprázdný řetězec.

Pravděpodobně nastane situace, kdy se budou dva uživatelé jmenovat stejně. Pravděpodobnost je větší v aplikaci *Chcu víno!*, jelikož se bude jednat o běžná jména, příjmení nebo přezdívky. Při vývoji bylo třeba zvážit, jestli tento možný problém řešit a nebo ne.

Z povahy aplikace nevypadá, že by vadilo mít duplicitní jména uživatelů. Aplikace zprostředkovává především navázání krátké komunikace mezi vinařem a zákazníkem. Vi-



Obrázek 5.1: Diagram znázorňující průběh přihlášení / registrace uživatele do aplikace.

naře zajímají především zprávy a poptávky z posledních hodin a dnů. Pravděpodobnost, že by v tomto krátkém období obdržel zprávy od více uživatelů se stejným jménem, je zanedbatelná. Také lze uvažovat, že pro vinaře je jméno druhotný údaj, důležitější je obsah zpráv.

Na druhou stranu se může stát, že dvě vinařství budou mít stejný název. Avšak lze předpokládat, že v každé vinařské oblasti budou jména unikátní, jelikož se přirozeně chtějí odlišit a jasně identifikovat.

Na základě výše uvedeného a po konzultaci s vedoucím mé práce jsem se rozhodl, že ponecháme uživateli možnost zvolit si libovolné (i duplicitní) jméno. Jestli to bylo správné rozhodnutí, se ukáže až po delším období na základě zpětné vazby uživatelů.

5.1.2 Odhlášení a odstranění účtu

Přihlášený uživatel má možnost se kdykoliv odhlásit i odstranit svůj účet. Obě akce se dají vyvolat v aktivitě *Můj profil*. Pro odhlášení je i rychlá akce v menu. Po odhlášení je uživatel odhlášen v aplikaci od *Google* účtu a přechází na úvodní obrazovku.

Pokud si uživatel přeje odstranit účet, tak je průběh totožný s odhlášením, ale současně jsou nevratně odstraněna veškerá data o uživateli (včetně vytvořených poptávek, soukromých zpráv, atd.) z databáze. Při následném přihlášení pomocí stejného účtu, je vytvořena nová entita v databázi a uživatel musí znovu vyplnit všechna povinná data.

5.2 Problémy objevené při implementaci

V následujícím textu jsou popsány některé problémy, které bylo třeba řešit při vývoji. Zaměření textu je na zajímavé problémy, které nejsou typickým úskalím vývoje.

5.2.1 Omezené funkce databáze *Firestore*

Tato databáze je mladý projekt z rodiny služeb *Firebase* a stále je v *beta* verzi. Obsahuje několik omezení, která bylo třeba akceptovat při vývoji aplikace.

Nejzávažnější omezení je nemožné vyhledávání pomocí více jak jednoho sloupce s neekvivalentní podmínkou (matematické operace *menší*, *větší*, *větší rovno*, *menší rovno*). Dalším omezení, respektive chybou služby, je neúplné vyhledávání pomocí objektu `GeoPoint`. Vyhledání proběhne pouze na základě zeměpisné šířky, nikoliv délky. Dle mého názoru to souvisí s předešlým omezením.

5.2.2 Rozšíření *Butter Knife*

Při implementaci byla aplikace rozšířena o knihovnu *Butter Knife*[1]. Knihovna zjednodušuje práci s prvky ze šablony aktivity, tím se zpřehlední a usnadní kód. Porovnání ukázkové implementace akce po kliknutí na tlačítko je v následujícím fragmentu kódu 5.1 a 5.2.

```
public class MainActivity extends AppCompatActivity {
    protected void onCreate( Bundle savedInstanceState ) {
        super.onCreate( savedInstanceState );
        Button button = findViewById( R.id.search_location );
        button.setOnClickListener( new View.OnClickListener() {
            public void onClick( View view ) {
                // ... definice akce
            }
        });
    }
}
```

Výpis 5.1: Standardní nastavení akce po kliknutí na tlačítko

```
public class MainActivity extends AppCompatActivity {
    @OnClick(R.id.search_location)
    public void onSearch() {
        // ... definice akce
    }
}
```

Výpis 5.2: Nastavení akce po kliknutí na tlačítko pomocí Butter Knife

[1] Butter Knife – http://jakewharton.github.io/butterknife/

Omezení *Butter Knife*

Knihovna působí jako dobrý pomocník při vývoji. Projekt *Chcu víno!* se skládá ze dvou aplikací, které sdílí společný modul. Nedostatkem knihovny je, že ji není možné použít v modulu. Knihovna oznámí chybu, že prvek má mnohonásobnou implementaci a není jednoznačně identifikovatelný.

Závěr

Knihovna dokáže zpřehlednit zdrojový kód, proto byla využita. Bohužel v modulech je nepoužitelná, proto většina zdrojového kódu tuto knihovnu nevyužívá.

5.2.3 Asynchronní akce

Během běhu aplikace je voláno velké množství asynchronních akcí, zejména práce s databází. Modelovou situací je konverzace zpráv. V konverzaci je nutné získat entitu konverzace a také oba účastníky konverzace. Získávání těchto dat probíhá pomocí asynchronních akcí.

Právě v detailu konverzace potřebujeme i data o protistraně, proto je nutné vyčkat, než jsou načtena všechna data. Kontrolu nad těmito akcemi lze získat pomocí knihovny *JDeferred*[2]. Knihovna vytvoří požadavek `Promise` („slib“ v doslovném překladu), který může být splněn nebo nesplněn.

S využitím této knihovny lze zobrazit grafiku znázorňující načítání (*progress bar*) obsahu. Jakmile jsou data načtena, tak je splněn „slib“ a lze zobrazit příslušná data nebo vyvolat akci. Pokud data nejsou načtena, tak vývojář zobrazí chybu, přesměruje na jinou aktivitu apod. V aplikaci je využito zanořování kontroly nad asynchronními akcemi, a proto hlavní „slib“ je splněn, jakmile jsou načteny všechny závislosti.

Každému „slibu“ lze specifikovat typ návratové hodnoty při úspěchu i neúspěchu. Pokud je třeba sledovat i průběh, tak se využije metoda `progress`. Základní ukázka kódu je v následujícím fragmentu kódu 5.3.

```
Deferred<Winery, Exception, Integer> deferred = new DeferredObject<>();
Promise<Winery, Exception, Integer> promise = deferred.promise();
promise.done( new DoneCallback<User>() {
    public void onDone( Winery result ) {
        hideProgressBar();
        // ... implementace akce
    }
}).fail( new FailCallback<Exception>() {
    public void onFail( Exception result ) { // ... implementace akce }
}).progress(new ProgressCallback() {
    public void onProgress(Integer progress) { // ... implementace akce }
});
showProgressBar();
getUser( deferred ); // Zavolání metody s asynchronní akcí
```

Výpis 5.3: Ukázka čekání na asynchronní akci s využitím knihovny *JDeferred*. První je vytvořen „slib“, kterému jsou nastaveny akce. Následně provedena akce na kterou se čeká.

[2]Knihovna je inspirována jQuery objektem Deferred. Oficiální dokumentace: https://github.com/jdeferred/jdeferred

5.2.4 Zobrazování detailu entity

V aplikaci je potřeba zobrazovat detail určité entity, například konverzace. Je potřeba předat aktivitě parametr s identifikátorem dané entity. Přepnutí do aktivity detailu probíhá pomocí záměru (`Intent`). Tomuto záměru je předán parametr s identifikátorem entity. Aktivita parametr zpracuje a spustí příslušný kód. Následující zjednodušený fragment kódu (5.4) demonstruje předání identifikátoru entity při zobrazení detailu konverzace.

```
/**
 * Třída aktivity přehledu konverzací
 */
public class ConversationListActivity extends AppCompatActivity {
    // Metoda po kliknutí na konverzaci v seznamu
    @Override
    public void onConversationClick(Conversation conversation, View view) {
        Intent i = new Intent( this, ConversationActivity.class );
        i.putExtra( ConversationActivity.EXTRA_CONVERSATION_ID,
            conversation.getId() );
        startActivity(i);
    }
}

/**
 * Třída aktivity detailu konverzace
 */
public class ConversationActivity extends AppCompatActivity {

    public static final String EXTRA_CONVERSATION_ID =
        "cz.adamecmartin.vino.CONVERSATION_ID";

    @Override
    protected void onCreate( Bundle savedInstanceState ) {
        super.onCreate( savedInstanceState );

        // Získání ID entity konverzace
        String mConversationId = getIntent().getStringExtra(
            EXTRA_CONVERSATION_ID );
        if ( mConversationId == null ) {
            // Id nebylo načteno
        }

        // Pokračování v aktivitě, načtení entity, ...
    }
}
```

Výpis 5.4: Předání identifikátoru záměru. Při vytváření záměru je objektu předán extra parametr s identifikátorem konverzace. Po vytvoření aktivity je tento parametr zpracován a ovlivňuje následné chování.

5.3 Vyhledávací proces v aplikaci Chcu víno!

V kapitole 4.7.1 návrhu GUI byl již vysvětlen smysl ovládacích prvků na úvodní obrazovce aplikace *Chcu víno!*, které slouží pro vytvoření vyhledávání zákazníky. Uživatelské rozhraní Vytvoření vyhledávání prošlo vývojem na základě uživatelského testování, viz kapitola 6.2. V nadcházejícím textu je popsán způsob implementace jednotlivých prvků vyhledávání.

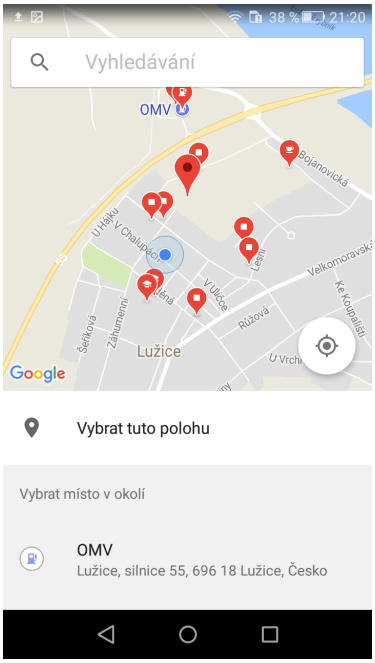
Cíl hledání

Tento parametr je reprezentován souřadnicemi bodu na mapě. Pravděpodobně nejčastěji bude tento bod totožný s aktuální polohou uživatele, proto se nabízí nastavit uživatelovu polohu jako výchozí. Bohužel *Android* často neposkytuje na první dotaz přesnou polohu a chvíli trvá, než získá skutečnou. Aby se předešlo využití nepřesné polohy uživatele, je uživatel povinen vybrat svou polohu, respektive cíl hledání.

Výběr polohy probíhá prostřednictvím speciální aktivity *Place Picker*[3]. Jedná se modul přímo od vývojářů OS *Android* a je implementován pomocí Google map. V aktivitě uživatel vidí svou polohu, lze vyhledat libovolnou adresu a jsou zobrazena místa v okolí, viz obrázek 5.2. Aplikace obdrží vybrané místo na mapě a má k dispozici souřadnice, adresu a název místa.

Uživateli je ve vyhledávacím formuláři zobrazen název místa. Pokud se jedná o známé místo, tak je uvedeno například „Vysoké učení technické Brno – Fakulta informačních technologií“, jinak souřadnice (obrázek 5.3).

Původně bylo zamýšleno využití adresy jako jednoznačného identifikátoru místa. Pokud uživatel zvolí polohu mezi dvěma obcemi (cyklostezka v lese), tak je zvolena adresa nejbližšího bodu nebo obce. Je pravděpodobné, že uživatelé budou využívat aplikaci i mimo přesně adresovatelná místa, proto se využívá popis souřadnicemi.



Obrázek 5.2: Ukázka aktivity pro vybrání cíle hledání. Je využita komponentu *Place Picker*.

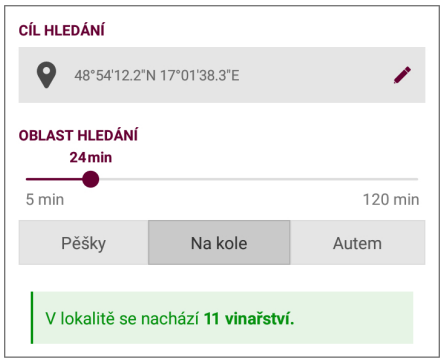
[3]Place Picker – https://developers.google.com/places/android-sdk/placepicker

Oblast hledání

Určuje maximální vzdálenost vinařství od cíle hledání. Je využita relativní jednotka – minuty, viz návrh aplikace kapitola 4.7.1. Počet minut je vybírán pomocí posuvníku *IndicatorSeekBar*[4], který je nadstavbou standardního posuvníku *SeekBar*[5] od tvůrců *Androidu*. Tento modul nabízí přizpůsobení počáteční a koncové hodnoty, možnost zobrazit a přizpůsobit indikátor aktuálně zvolené hodnoty a mnoho dalšího. Výsledná podoba posuvníku je na obrázku 5.3.

Uživatel si vybírá, jakým způsobem plánuje vzdálenost překonat. Pro každý způsob byla stanovena konstanta průměrné rychlosti, pomocí níž se vypočítá rádius hledání.

- **Pěšky** – 4.2 km/h
- **Na kole** – 15 km/h
- **Autem** – 45 km/h



Obrázek 5.3: Grafická reprezentace zadávání parametrů hledání v aplikaci Chcu víno!

Zpráva pro vinaře

Posledním parametrem je zpráva pro vinaře, kde může zákazník specifikovat svou poptávku. Na základě testování (kapitola 6.2) byla editace výchozí zprávy přesunuta do mezikroku, před vytvořením hledání. Je zadávána pomocí dialogového okna.

Vytvoření hledání

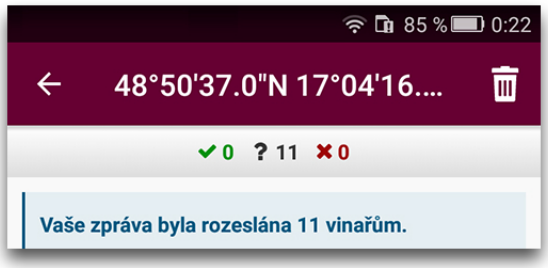
Jakmile jsou výše zmíněné parametry vytvořeny, je vyhledávání uloženo do databáze. Pomocí funkcí ve službě *Firebase Functions*, která byla popsána v kapitole 4.2.2, jsou spuštěny další akce. Prvním krokem je nalezení všech vinařů v hledané oblasti. Těmto vinařům je v databázi vytvořena konverzace spjatá s hledáním, představují poptávku. Po vytvoření poptávky je uživateli automaticky zaslána notifikace pomocí *Firebase Functions*.

Po vytvoření poptávek pro všechny vinaře v oblasti, je k hledání uložena informace, kolika vinařům byla poptávka zaslána. Zákazník tuto informaci vidí v detailu hledání.

Při přijmutí poptávky vinařem, je zákazník upozorněn notifikací a současně se mu aktualizuje seznam vinařů, kteří přijali poptávku. Při zamítnutí není uživatel nijak upozorněn,

[4]IndicatorSeekBar – https://github.com/warkiz/IndicatorSeekBar
[5]SeekBar – https://developer.android.com/reference/android/widget/SeekBar

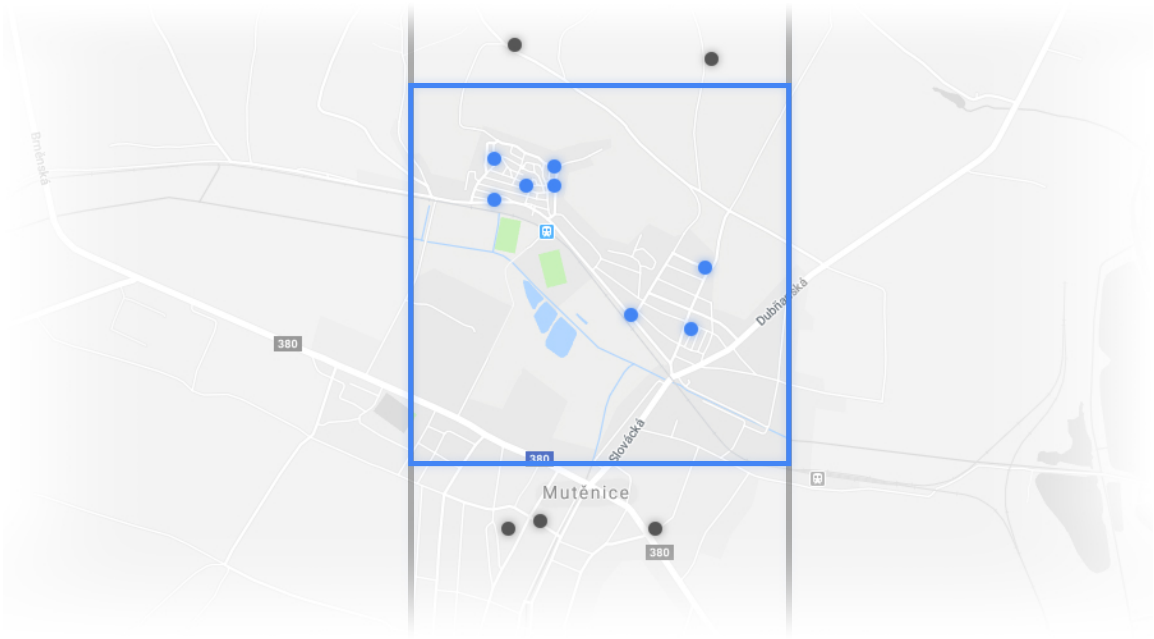
pouze se mu aktualizuje počet zamítnutých a čekajících poptávek. Statistika přijatých, čekajících a zamítnutých poptávek je uživateli neustále zobrazena v záhlaví detailu hledání (obrázek 5.4).



Obrázek 5.4: Rychlý přehled počtu přijatých, čekajících a zamítnutých poptávek v záhlaví hledání.

Problém při vyhledávání v databázi dle souřadnic

Vinaři jsou vyhledáni dle prohledávané oblasti. Oblast představuje čtverec, kde rohy jsou reprezentovány souřadnicemi bodu na mapě. Bohužel aktuální verze *Firestore* dokáže vyhledat pouze na základě zeměpisné šířky. Z databáze se tedy získá zleva a zprava omezená množina bodů představující vinařství. Následně programátor musí zajistit omezení shora a zdola. Situaci demonstruje následující obrázek 5.5, kde šedé body značí vinařství, která jsou nalezena databází, i když nejsou v hledané oblasti. Naopak modré body značí vinařství, kterým bude oprávněně rozeslána poptávka.



Obrázek 5.5: Demonstrace chyby vyhledávání ve *Firestore* dle souřadnic v objektu `GeoPoint`. Z databáze jsou vrácena vinařství omezená pouze zeměpisnou šířkou. Zeměpisná délka není zohledněna a vývojář je nucen udělat dodatečnou selekci.

Možným řešením bylo rozštěpit souřadnice z objektu `GeoPoint` do dvou hodnot, první pro zeměpisnou šířku, druhá pro zeměpisnou délku. Při pokusu o toto řešení bylo nalezeno další omezení databáze. *Firestore* nedokáže neekvivalentně vyhledávat pomocí více jak jednoho parametru. Tento problém je popsán v předešlé kapitole 5.2.1.

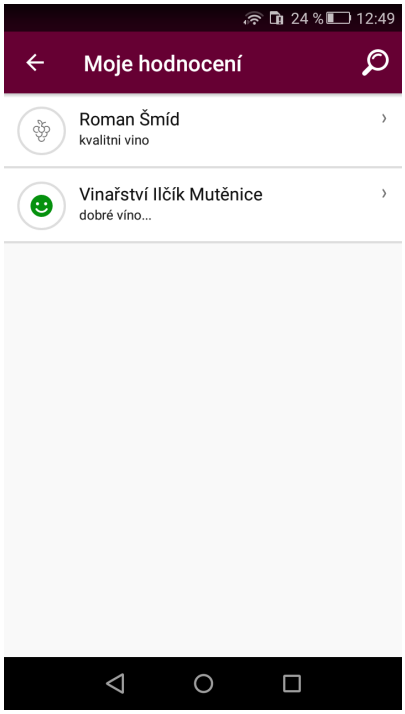
5.4 Soukromé hodnocení uživatelů

Při analýze práce bylo zjištěno, že zákazníci by mohli ocenit soukromé poznámky k vinařů a opačně. Zákazník si vloží poznámku k vinaři, který má dobré víno. Napíše si, které víno mu chutnalo. Příště ví, které víno a vinaře zvolit. Naopak, pokud mu nechutnalo nebo má negativní zkušenost s přístupem, už tam příště nepůjde. Na druhou stranu vinař si udělá poznámku, že zákazník nepřišel, i když se domluvili na ochutnávce vín. Příště si dá u něj pozor, případně mu nenabídne své služby.

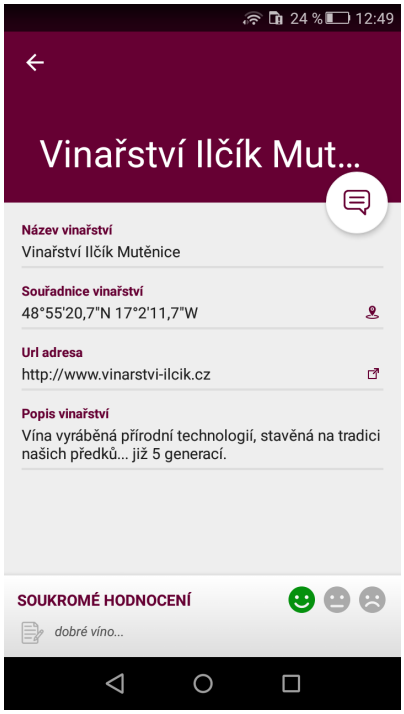
Vedle textového hodnocení bylo implementováno i grafické. Uživatel si zvolí ze tří emotikon, značící pozitivní, neutrální a negativní zkušenost. Podoba ikon je na obrázku 5.7.

Aby uživatel nemusel dohledávat hodnocení, je grafické hodnocení zobrazeno napříč aplikací. Uživatel vidí v seznamu zpráv, vinařů či poptávek příslušnou ikonu. Ihned ví, jestli s daným uživatelem již měl zkušenost a jak ji hodnotil (obrázek 5.8). Pro přehled všech hodnocení uživatele byla vytvořena aktivita `RatingActivity`. Aktivita vypisuje seznam udělených hodnocení abecedně dle jména uživatele. Pro lepší komfort uživatele je možné v seznamu vyhledávat. Náhled aktivity je na obrázku 5.6.

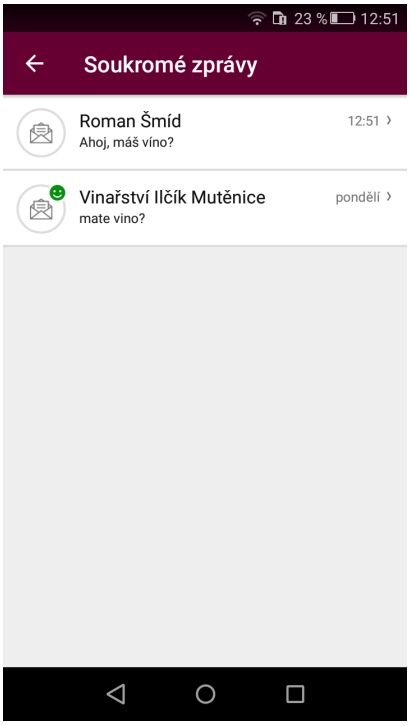
Uživatel zadává hodnocení na profilu hodnoceného uživatele. Box s hodnocením je umístěný v dolní části obrazovky a je stále viditelný (obrázek 5.7).



Obrázek 5.6: Aktivita zobrazuje seznam hodnocení. Zákazník vidí svou poznámku k vinařství a také zvolenou emotikonu.



Obrázek 5.7: Zadávání hodnocení a volba emotikony probíhá na profilu hodnoceného uživatele.



Obrázek 5.8: Grafické hodnocení (emotikon) je zobrazeno napříč aplikací, například v seznamu konverzací.

Veřejné hodnocení

Hodnocení není veřejné. Poznámka slouží pro uživatelovu potřebu. Aplikace má cíl být privátním pomocníkem, nikoliv veřejným srovnavačem vinařů. Kdyby hodnocení bylo zveřejněno, vzniklo by riziko cíleného poškozování vinařů a bylo by vhodné filtrovat vulgarismy.

Avšak veřejné hodnocení by uživatelům mohlo pomoct při výběru vinaře. V dalším vývoji aplikace bude zváženo, vymyšlení vhodné formy veřejného hodnocení.

5.5 Zveřejnění aplikace

Jakmile byly aplikace připraveny na první testování uživateli, bylo třeba je zveřejnit. Zveřejnění proběhlo v on-line distribuční službě Google Play[6]. Služba slouží především k distribuci aplikací pro mobilní zařízení s operačním systémem *Android*.

V administračním prostředí této služby je možné zvolit několik způsobů zveřejnění.

Alfa kanál Určený pro první testování, většinou menší skupinou testerů, například zaměstnanci vývojářské firmy. Vývojář zadává seznam testerů (pomocí e-mailových adres).

Beta kanál Jako další krok je doporučeno využití tohoto kanálu, který je určený na testování větší skupinou uživatelů. Opět je na vývojáři, komu povolí přístup.

Produkční kanál V tomto kanálu je veřejná verze aplikace (přístupná všem).

[6]Google Play – https://play.google.com

První verze aplikací byly zveřejněny v alfa kanále. Následně byly opraveny chyby odhalené při testování a zapracovány věcné připomínky ke grafické podobě aplikace. Po otestování úprav byly aplikace vydány v produkčním kanále a tím zpřístupněny široké veřejnosti. Více o testování aplikací je uvedeno v kapitole 6, zabývající se testováním. Následují odkazy pro stažení aplikací v Google Play:

- **Chcu víno!** – play.google.com/store/apps/details?id=cz.adamecmartin.chcuvino
- **Mám víno!** – play.google.com/store/apps/details?id=cz.adamecmartin.mamvino

5.6 Prezentace a propagace aplikací

Aplikace je vyvíjena pro širokou veřejnost. Aby se mohla dostat k uživatelům, je třeba ji prezentovat a šířit. Jak bylo uvedeno v analýze projektu, kapitola 2.1, vhodná volba propagace je jedno z úskalí aplikace.

Způsoby propagace jsou rozděleny do několika kategorií, které jsou níže popsány. Všechny způsoby se snaží dodržovat podobné principy, především stále stejnou barvu aplikace a často opakující se logo, aby se lépe vrylo uživatelům do paměti.

5.6.1 Webové prezentace

Pro jednoduší představení projektu byly vytvořeny webové prezentace. Každá z aplikací má vlastní doménu, která je v barvách aplikace. Obsahem je stručné vysvětlení funkčnosti a odkaz ke stažení.

Prezentace pro aplikaci *Chcu víno!* se nachází na adrese chcuvino.cz. Analogicky aplikace *Mám víno!* má prezentaci na adrese mamvino.cz.

5.6.2 Sociální sítě

Většina moderních aplikací je doprovázena profilem na sociálních stránkách. Prozatím byly vytvořeny stránky na sociální síti *Facebook*[7]. Každé stránce je možnost vytvořit unikátní přezdívku, která reprezentuje stránku a je využita v URL. Pro aplikaci *Chcu víno!* je profil na URL adrese facebook.com/**chcuvinocz** a pro *Mám víno!* facebook.com/**mamvinocz**.

5.6.3 Reklama

Způsobů reklamy je nespočet. Pro aplikace bylo třeba najít finančně dostupný způsob, jelikož aplikace není monetizována a nebyla by možná návratnost investice. Zavržena byla možnost masivní reklamy (např.: v rádiu nebo televizi). Další specifikem je, že každá z aplikací má jiný cíl pro uživatele, proto se propagace částečně liší.

Reklama pro Chcu víno!

Aplikaci má smysl prezentovat pouze v oblastech, kde je zaregistrovaných několik vinařů v aplikaci *Mám víno!*. Zákazníci v oblasti tráví typicky více dnů a využívají možnost ubytování v okolních apartmánech, ubytovnách nebo kempech.

Z výše uvedených důvodů, byla zvolena propagace pomocí reklamních letáků. Letáky se můžou umístit do ubytovacích zařízení, vinných sklepů apod. Vytvořené letáky jsou

[7] https://facebook.com

k náhledu v příloze B. Důraz byl kladen na vytvoření výrazného letáku, byla využita hlavní barva aplikace. Leták je doplněn úvodní obrazovkou aplikace společně s popisem. Slouží i jako stručný návod.

Letáky jsou vytvořeny ve velikosti formátu A5 (148 mm x 210 mm) a jedna z variant ve velikosti standardní poštovní obálky.

Reklama pro Mám víno!

Nejefektivnější způsob získávání nových zákazníků je přes osobní kontakt. Nejlépe za pomoci místního člověka, tedy osoby, které důvěřuje potencionální uživatel. Současně by měla tato osoba důvěřovat smyslu aplikace. Obecná reklamní kampaň, bez osobního kontaktu, má smysl až v momentu, kdy bude aplikace úspěšně zavedena v prvních vinařských oblastech.

Aby vinař nezapomněl název aplikace či její smysl, byl vytvořen leták podobný tomu, jako pro *Chcu víno!*, viz příloha B. Leták se dá rozdat také na pravidelných schůzích vinařského spolku nebo vinařských akcích.

5.6.4 Osobní propagace

Moje rodina pěstuje hrozny již několik generací a známe se s velkým množstvím vinařů. První kroky propagace vedou přes tyto vinaře. Tito vinaři můžou pomoct s propagací aplikace v jejich vinařské oblasti.

5.7 Rizika zneužití aplikace

Při návrhu a implementaci byla zhodnocena rizika zneužití aplikace. Nejjednodušší zneužití může být **rozesílání nevyžádaných reklamních sdělení** (SPAM). V aktuální verzi aplikace není implementována žádná ochrana proti tomuto způsobu zneužití. Po nějaké době provozu aplikace lze zanalyzovat databázová data a zpětnou vazbu uživatelů, jestli byla aplikace zneužita pro rozesílání SPAM-u.

V případě opakovaného vytváření vyhledávání (v krátkém sledu) zákazníkem, kde nereaguje na přijmutí vinařem, lze zavést minimální rozestup mezi novým a posledním vyhledáváním. SPAM může být odhalen i v soukromých zprávách, prevencí je přidání možnosti zablokovat uživatele.

Kapitola 6

Testování

Vyvinutím první verze aplikace nekončí práce, ale začíná jedna z nejdůležitějších částí a to je testování. V následujícím textu bude čtenář seznámen s provedenými testy v průběhu vývoje a po jeho dokončení.

Testování proběhlo v několika fázích projektu, různými způsoby a v několika iteracích.

6.1 Testování návrhu GUI

Po vytvoření skic GUI (viz kapitola 4.7) bylo třeba ověřit, zda návrh je srozumitelný pro nestranného uživatele. Testování proběhlo na 3 osobách. Jako způsob testování byl zvolen následující postup. Uživatel dostal postupně k nahlédnutí skici, v pořadí totožném se skutečným procesem aplikace. Úkolem bylo slovní pojmenování významu ovládacích prvků, následně byla vysvětlena úloha dané skici a testovaný uživatel poskytl chybějící či přebývající elementy.

Jelikož testování probíhalo slovní formou, nelze získané údaje převést do grafické reprezentace, jako je graf či tabulka. Níže jsou uvedeny připomínky, které byly zohledněny při editaci skic a funkčnosti.

- Přidání informace do detailu poptávky, co se aktuálně děje, viz obrázek 4.7. Chvíli trvá, než se poptávka rozešle vinařům a především reakce vinařů je několik minut až hodin. Dále se přidalo zobrazení, kolik vinařů obdrželo poptávku a kolik z nich se již vyjádřilo.
- Přidání historie poptávek (slouží ke zpětnému dohledání vinařů).

Závěr z testování návrhu

Výsledek testu byl pozitivní. Testované osoby většinu ovládacích prvků pochopily správně. Zejména vstupní bod aplikace byl pochopen bez poznámek, což byl hlavní důvod testování. Je nutno vzít v potaz, že to testovaly pouze tři osoby a tedy jejich názory nelze brát obecně, nicméně další testování bude probíhat již v částečně funkční aplikaci.

V průběhu implementace byl upraven vzhled aplikace oproti skicám, proto i další testování je většinově zaměřeno na GUI.

6.2 Testování první verze

První verze aplikace na *Google Play* byla vydána v alfa kanále, který je určen pro první testování. Testování se účastnilo 8 lidí, kteří měli za úkol si aplikaci stáhnout a klikat na vše, co je napadne. Čímž byla snaha o odchycení logických chyb zdrojového kódu.

Následně měli za úkol vytvořit vyhledávání a projít si celým procesem hledání, tedy komunikace s vinaři apod. Nakonec bylo krátké slovní zhodnocení aplikace. Pozitivní bylo, že aplikace *Mám víno!* byla bez připomínek. Objevily se však nedostatky v aplikaci *Chcu víno!*.

Zpráva vinařům

Vyplnění zprávy pro vinaře mělo být jako třetí prvek na úvodní obrazovce, nicméně ve zpětné vazbě bylo 2x zmíněno, že si testovaná osoba možnosti vyplnění zprávy nevšimla.

Řešení: Zadání zprávy přesunuto do druhého kroku vytváření hledání. V tomto kroku se objeví dialogové okno, kde uživatel může a nemusí změnit výchozí zprávu. Po vyplnění zprávy je vytvořeno hledání.

Informace, kolik vinařů se v hledané oblasti nachází

Když byly zadány parametry lokality a oblasti hledání, uživateli nebylo sděleno, kolik vinařů bylo v dané lokalitě nalezeno. Tím vznikali hledání, které nenašlo žádného vinaře a bylo pro uživatele ztrátou času.

Řešení: Byl přidán informační box nad potvrzovací tlačítko (*Vyhledat*), viz obrázek 6.1. V boxu je napsán počet nalezených vinařů ve zvolené oblasti. Pokud nebyl nalezen žádný vinař, tak je uživatelovi znemožněno vytvořit hledání.

Text vyhledávacího tlačítka

Text *Chcu víno* byl špatně chápán a nedostatečně představoval tlačítko. Text byl nahrazen za *Vyhledat* a grafika tlačítka byla upravena (viz obrázek 6.1).

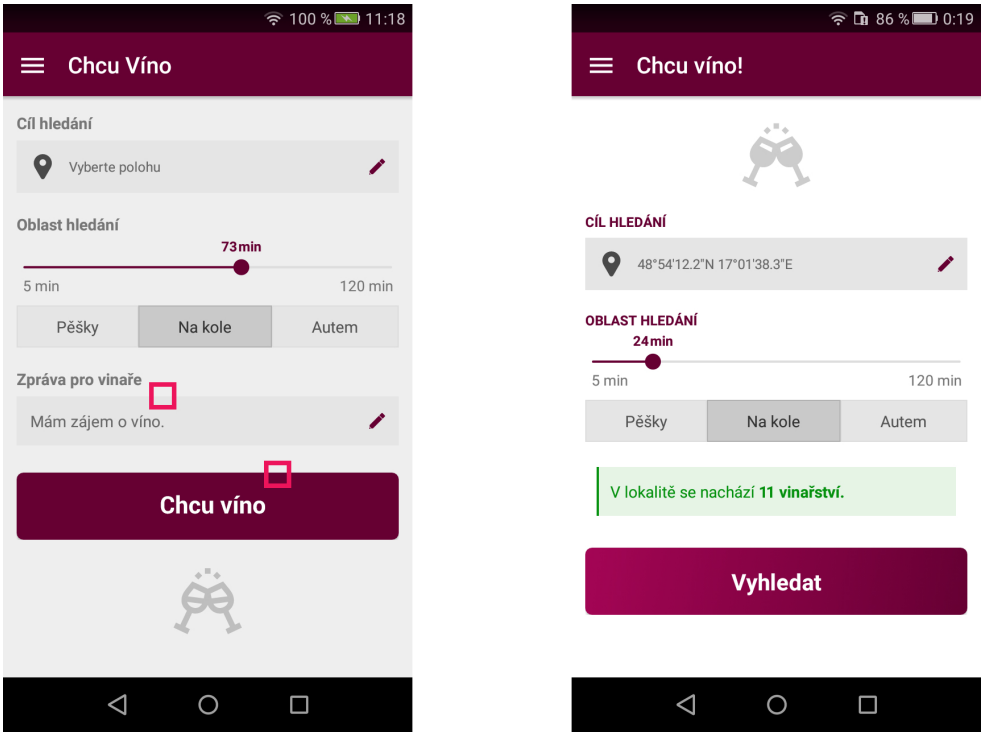
Detail poptávky

Nedostatečné informace, co se právě děje. Bylo třeba zanalyzovat všechny možné stavy (zpráva se rozesílá vinařům, čekání na reakci vinařů, vinaři již reagovali, hledání vypršelo, atd.) a předat uživateli odpovídají informaci. Statistika vyhledaných/potvrzených/zamítnutých vinařů byla umístěna v dolní části obrazovky a uživatelé si toho nevšimli, proto byla přesunuta do vrchní části. Porovnání obrazovky před a po testování je v obrázku 6.2.

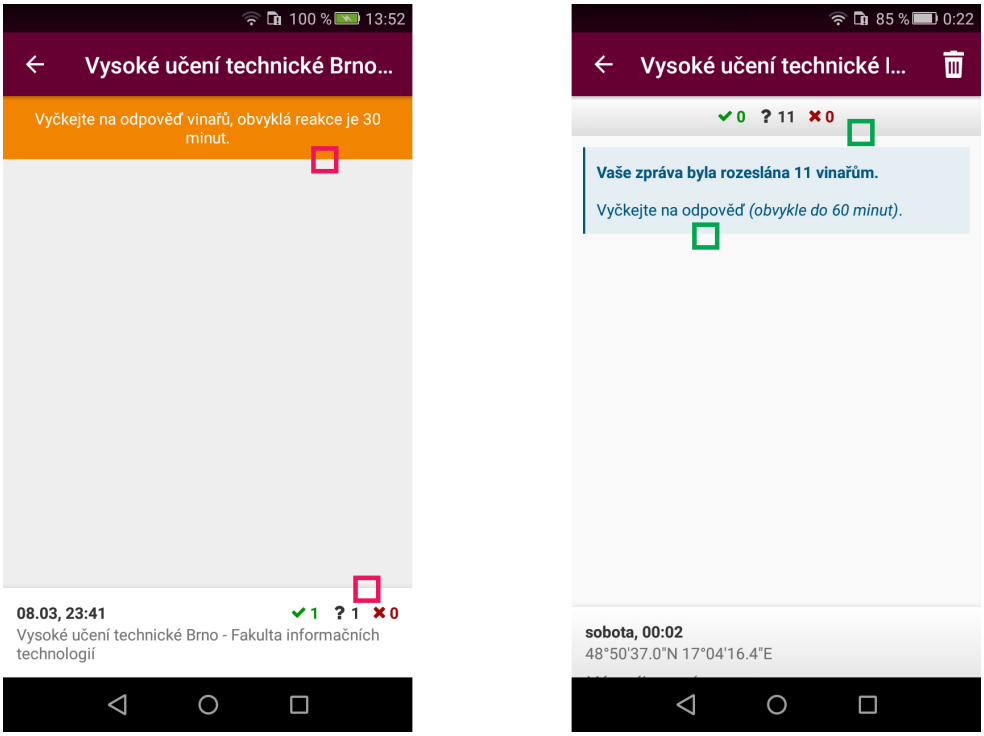
Závěr z testování

Aplikace se skládá z relativně malého počtu obrazovek. Testeři se rychle zorientovali v navigaci. Ovšem problém byla absence dostatečně podrobných informací, které uživateli pomůžou se orientovat.

Avizované nedostatky byly odstraněny a testeři znovu prošli procesem poptávky, tentokrát bez výhrad.



Obrázek 6.1: Porovnání úvodní obrazovky před (vlevo) a po testování. Růžové čtverce značí odhalené nedostatky, které byly upraveny.



Obrázek 6.2: Porovnání obrazovky poptávky před (vlevo) a po testování. Růžové čtverce značí odhalené nedostatky, které byly upraveny (zelené čtverce).

6.3 Testování různých verzí OS

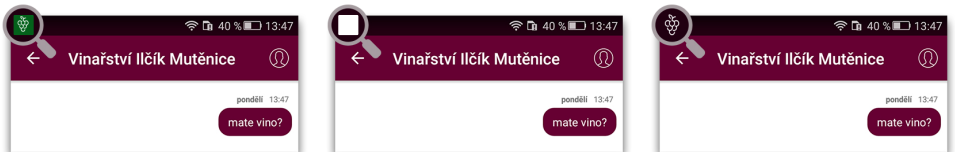
Nedílnou součástí mobilní aplikace je testování na nejrůznějších verzích operačního systému *Android*. Minimální verze je 5.0 *Lollipop*. Aplikace byl postupně otestována na verzi 5.0, 5.1, 6.0 *Marshmallow*, 7.1 *Nougat* a 8.0 *Oreo*, která je poslední vydanou.

Během testování se nevyskytla žádná závažná chyba, tedy například situace, kde aplikace se musí chovat jinak na základě verze OS. Byly odhaleny dva nedostatky, kde jeden byl závažnější.

Ikony notifikací

Od verze OS 6.0 *Marshmallow* je nutné mít ikonu pro notifikace pouze v bílé barvě. Pokud tomu tak není, zobrazí se pouze bílý čtverec. Chyba je znázorněna na následujícím obrázku 6.3.

Řešení: Původní ikony notifikací byly upraveny pouze do bílé barvy.

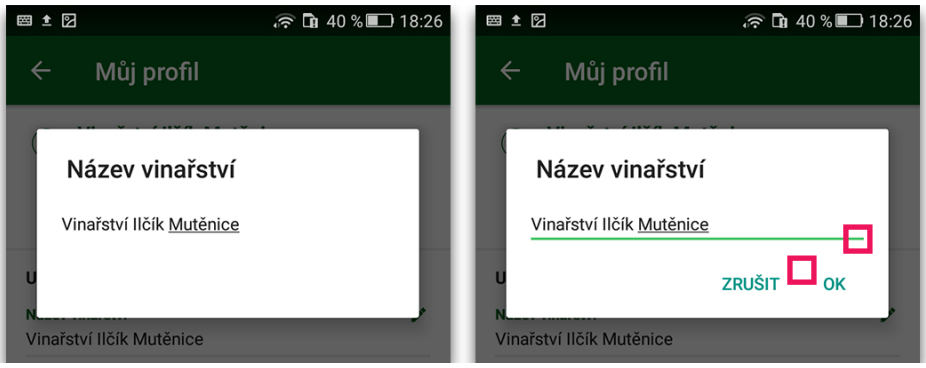


Obrázek 6.3: Vývoj podoby ikony notifikací. Vlevo je původní barevná. Uprostřed stejná ikona v OS verze 6 a vyšším. Napravo upravená ikona, kompatibilní se všemi verzemi OS.

Nesoulad barevného schématu

Špatnou volbou barev prvků, jako je sekundární barva *colorAccent*, nastal problém např. při editaci informací o uživateli v dialogu, viz obrázek 6.4. Kde nebylo vidět tlačítko pro potvrzení nebo podtržení textového pole.

Řešení: Nastavení správných barev a vyhnutí se přepisování barev pro dialog. Na základě toho se uživateli vždy zobrazovaly správné (kontrastní) barvy. Pokud vývojář chce změnu barev i pro dialog, je nutné tak provést explicitně ve vytváření tohoto dialogu.



Obrázek 6.4: Chyba v nastavení barev aplikace. Na pravém obrázku lze vidět chybějící prvky z levého obrázku, které byly bílou barvou.

6.4 Testování produkční verze

Jakmile skončilo uzavřené testování, byly aplikace vydány v *Google Play* pro všechny. Aplikace musely být otestovány od nezasvěcených uživatelů. Problémem bylo, jak aplikace dostat k těmto uživatelům.

Jako vhodný způsob byla zvolena forma letáků. Letáky bylo třeba vhodně navrhnout. Výsledek měl být poutavý na první pohled. Do letáku bylo umístěno logo, název a podtitul. Zároveň měl vysvětlit fungování aplikace, proto byly transformovány informace z webových stránek do stručnější formy a umístěny do letáku. Pro lepší představu o fungování aplikace byl přidán náhled úvodní obrazovky.

Leták je doplněn sloganem, aby byl opravdu poutavý. Slogan v pár slovech vystihuje smysl aplikace. Pro aplikaci *Chcu víno!* byly zvoleny dva slogany a pro *Mám víno!* jeden.

- *„Poznejte vína a atmosféru malých vinařství bez klepání.“* – Chcu víno!
- *„Najděte vinaře, který má na Vás čas.“* – Chcu víno!
- *„Nabídněte svá vína zákazníkům pohodlně a zdarma.“* – Mám víno!

Výsledná podoba letáků je zobrazena v příloze B. Leták byl doplněn QR kódem, pro pohodlnější instalaci aplikace, adresou webové prezentace a Facebook účtu.

Distribuce letáků

Testovací oblastí byla zvolena vinařská obec *Mutěnice*. Jedná se o vyhlášenou oblast, kterou každé léto navštíví tisíce turistů. Současně se zde nachází i vinný sklep našeho rodinného vinařství, včetně sídla.

Letáky byly umístěny do 6 penziónů / ubytovacích zařízení, do dvou hospod a jedné cukrárny. Zmíněná zařízení obdržela především letáky pro aplikaci *Chcu víno!*, jelikož na těchto místech se zdržují návštěvníci. Provozovatelé dostali i leták pro *Mám víno!*, protože většina z nich vlastní i vinařství.

Letáky aplikace *Mám víno!* byly rozdány několika vinařům osobně. Navíc byly předány představiteli vinařského spolku Mutěnice, který aplikaci na schůzi prezentoval a všem zúčastněným členům rozdal propagační leták.

6.4.1 Měření úspěšnosti jednotlivých letáků

Každý typ letáku obsahoval QR kód lišící se v cílové adrese. Adresa první směřuje na webovou prezentaci, kde se zaznamená, z jakého typu letáku uživatel přišel a následně je přesměrován do obchodu *Google Play*. Na základě těchto informací lze vytvořit statistiku úspěšnosti jednotlivých sloganů. Měření je pouze orientační, někteří uživatelé nevyužijí QR kód. Současně je nutná spolupráce zařízení, které obdržely letáky. Některé zařízení pouze umístí letáky například na recepci, jiné aktivně nabídnou leták hostům.

Data z testování

Testování započalo na konci vývoje aplikace, jen několik týdnů před odevzdáním práce. Je třeba zohlednit, že to bylo v měsíci dubnu a květnu, kdy není hlavní turistická sezóna.

Aplikace Mám víno!

Do aplikace bylo registrováno 8 vinařů (tabulka 6.1) z oblasti obce Mutěnice, z celkového počtu asi 25 oslovených. Vinaři poukazovali na problém vystavování dokladů zákazníkovi. Malému vinaři se nevyplatí pořizovat si EET pokladnu, kvůli stovkám litrů, které by ročně prodal. Takovou překážkou bohužel nejde v rámci aplikace odstranit. Celkově převládal názor, že aplikace může být přínosná, ale velká část vinařů nechce investovat čas do něčeho, co zaručeně nepřinese nové zákazníky, respektive zisk.

Aplikace Chcu víno!

Hlavní aplikace za první týdny mnoho zájemců nepřilákala. Po odfiltrování vinařů, kteří si i tuto aplikaci ze zajímavosti nainstalovali, zbyly pouze dva uživatelé. Nicméně aplikaci nevyužili (nevytvořili hledání). Může to znamenat, že v hledané oblasti nebyl žádný vinař, proto vyhledání se nepodařilo vytvořit. Ovšem může za tím být i nepochopení ovládacích prvků nebo smyslu aplikace. Celkově byla aplikace nainstalována 6-ti uživateli (tabulka 6.1) a neproběhla žádná odinstalace.

Závěr z testování

Pozitivním výsledkem je zájem vinařů o aplikaci, bez čehož se aplikace nemůže rozvíjet. Další pozitivum je, že ani jedna z aplikací nebyla uživatelem odinstalována (dle statistik v *Google Play Console*) a nenastal žádný pád aplikace. Poptávky proběhly pouze zkušební mezi vinaři a kamarády. Notifikace se správně vytvářely a aplikace reagovala správně.

Pro statistické účely bude vhodné si zaznamenávat všechna hledání, tedy i ta neúspěšná (hledání nejde vytvořit, pokud v oblasti nebyl nalezen žádný vinař). Tato data bude možné využít pro stanovení pořadí zavádění (propagace) aplikací v konkrétních vinařských oblastí.

QR kódy byly využity ve třech případech, dvakrát kód z letáku B.3a a jedenkrát z letáku B.3b. Jestli zvolený způsob propagace byl správný ukáže až větší časový úsek. Taktéž plánované vytvoření aplikace pro operační systém *iOS* (*Apple*) by mohl využívání aplikace pozvednout.

Aplikace / Verze OS	5.0	5.1	6.0	7.0	7.1	8.1	Celkem
Mám víno!	1	1	1	3	1	1	**8**
Chcu víno!	0	0	1 (1)	2 (0)	1 (0)	2 (1)	**6 (2)**

Tabulka 6.1: Přehled verzí systému uživatelů aplikací. U aplikace *Chcu víno!* je napsán v závorce počet účtů, které nesouhlasí s účtem některého z vinařství.

Kapitola 7

Závěr

Cílem práce bylo vytvoření mobilní aplikace umožňující spojení zájemců o víno ve zvolené oblasti s vinaři. Zvolenou platformou pro vývoj byl operační systém (OS) *Android*, a to zejména z důvodu, že se jedná o nejpoužívanější mobilní OS. Na začátku práce je *Android* popsán po technické stránce.

Aplikace není pouhým vyhledávačem vinařů dle parametrů. Při vytvoření vyhledávání je vinařům odeslána notifikace s poptávkou. Vinař si vybere, jestli je dostupný nebo ne. Zákazník obdrží seznam vinařů, kteří mají opravdu čas se zákazníkovi věnovat. Obě strany si můžou ujasnit podrobnosti pomocí soukromých zpráv přímo v aplikaci. Ze seznamu si zákazník vybere jednoho vinaře či více vinařů, se kterými se domluví na návštěvě.

Mezi další funkce aplikace patří zobrazení seznamu vinařů, kteří jsou seřazeni dle vzdálenosti od uživatele. Skrze tento seznam se uživatel dostane na profil vinaře a může navázat komunikaci pomocí soukromých zpráv, navštívit webové stránky nebo mu přímo zavolat.

Pro zpříjemnění používání aplikace bylo implementováno soukromé hodnocení uživatelů. Uživatel nepotřebuje další nástroj pro poznamenání vinaře, který má dobré víno apod. Hlavní důraz byl kladen na grafické uživatelské rozhraní. Uživatel by měl vždy na obrazovce vidět hlavní informace, bez nutnosti přepínání do jiných aktivit. Při vývoji aplikace bylo prováděno iterativní testování. Na základě zpětné vazby se aplikace upravovala. Součástí práce bylo vytvoření videa demonstrujícího funkčnost aplikace a plakátu.

Zhodnocení výsledku práce

Požadavky na práci byly splněny, aplikace je v reálném provozu a má potenciál být využitelná. K aplikaci byly vytvořeny propagační letáky, které byly umístěny do ubytovacích zařízení ve vinařské obci Mutěnice. Letáky propagující aplikaci pro vinaře *Mám víno!* byly rozdány vinařům v této oblasti. Projekt je doprovázen i webovou prezentací a profilem na sociálních sítích.

7.1 Pokračování ve vývoji aplikací

Ve vývoji aplikací *Chcu víno!* a *Mám víno!* chci pokračovat i po odevzdání diplomové práce. Se zaměřením na odstranění nedostatků a chyb, které se objeví během prvních měsíců používání reálnými uživateli. Hlavní sezóna je během teplých měsíců, především letních prázdnin. Pouze dlouhodobé testování ukáže skutečný potenciál aplikace.

Mezi zvažované rozšíření funkcionality patří především obohacení profilu vinaře o komplexnější informace: seznam dostupných vín, fotogalerie, možnost mít více provozoven, na-

bídka ubytování a občerstvení. V aplikaci pro zákazníky bude třeba tyto informace vhodně zobrazovat. Všechny informace mají mít pouze informativní charakter, aplikace nemá sloužit jako internetový obchod.

Aplikaci pro zákazníky bude možné rozšířit o kalendář vinařských akcí, komplexnější poznámky (hodnocení konkrétních vín) a další.

Vytvoření aplikace pro *iOS*

Aktuální verze je pouze pro *Android* a to může být překážkou při rozšiřování aplikace. Druhým největším zástupcem operačních systémů v mobilních zařízeních je *iOS* od společnosti *Apple*. Primárním cílem bude vytvořit aplikaci pro *iOS* a následně budou zvažována vylepšení, viz výše.

Literatura

[1] *Android Architecture*. [Online; navštíveno 20.05.2018].
URL http://shubhamjain.space/android/android-architecture/

[2] *Android ContentProvider tutorial*. [Online; navštíveno 20.05.2018].
URL http://en.proft.me/2017/03/12/android-contentprovider-tutorial/

[3] *Cloud Firestore*. [Online; navštíveno 20.05.2018].
URL https://firebase.google.com/docs/firestore/

[4] *Dashboards*. [Online; navštíveno 20.05.2018].
URL https://developer.android.com/about/dashboards/index.html

[5] *Firebase dokumentace*. [Online; navštíveno 20.05.2018].
URL https://firebase.google.com/docs/guides/

[6] *Firebase Realtime Database*. [Online; navštíveno 20.05.2018].
URL https://firebase.google.com/docs/database/

[7] *Intents and Intent Filters*. [Online; navštíveno 20.05.2018].
URL https://developer.android.com/guide/components/intents-filters.html

[8] *Introduction to Android*. [Online; navštíveno 20.05.2018].
URL https://developer.android.com/guide/index.html

[9] *Material Design*. [Online; navštíveno 20.05.2018].
URL https://material.io

[10] *Material Design for Developers*. [Online; navštíveno 20.05.2018].
URL https://developer.android.com/training/material/index.html

[11] *Průzkum: Víno pijeme častěji a více za něj utrácíme!* [Online; navštíveno 20.05.2018].
URL https://www.wineofczechrepublic.cz/akce-a-novinky/aktuality/pro-tisk/6895-pruzkum-vino-pijeme-casteji-a-vice-za-nej-utracime.html

[12] *Smartphone OS Market Share*. [Online; navštíveno 04.09.2017].
URL https://www.idc.com/promo/smartphone-market-share

[13] *The Activity Lifecycle*. [Online; navštíveno 20.05.2018].
URL https://developer.android.com/guide/components/activities/activity-lifecycle.html

[14] *UX design: digitální alchymie dneška*. [Online; navštíveno 20.05.2018].
URL https://www.mibcon.cz/a/ux-design-digitalni-alchymie-dneska

[15] Luboslav Lacko: *Vývoj aplikací pro Android*. Computer Press, 2015, ISBN 978-80-251-4347-6.

[16] Norman, D. A.: *Design pro každý den*. Dokořán, 2010, ISBN 978-80-7363-314-1.

Příloha A

Obsah CD

- `/xadame36-DP.pdf` – elektronická verze písemné zprávy
- `/doc.zip` – komprimované zdrojové soubory písemné zprávy
- `/source-code.zip` – komprimovaný zdrojový kód pro *Android Studio*
- `/javadoc.zip` – komprimovaná dokumentace zdrojového kódu (`javadoc`)
- `/chcuvino.apk` – balíček pro systém *Android* aplikace *Chcu víno!*
- `/mamvino.apk` – balíček pro systém *Android* aplikace *Mám víno!*
- `/poster-DP.pdf` – plakát k aplikaci
- `/posters.pdf` – propagační letáky aplikací *Chcu víno!* a *Mám víno!*
- `/video.mp4` – doprovodné video k aplikaci

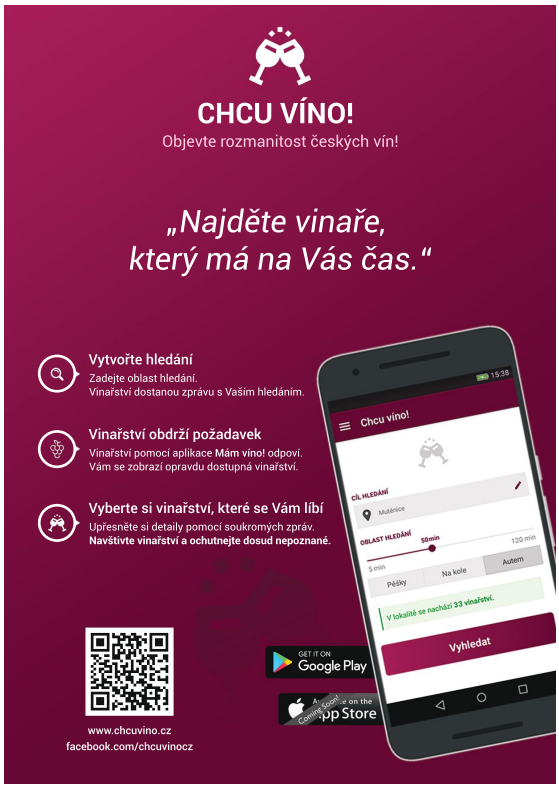
Příloha B

Propagační letáky

Obrázek B.1: Leták Mám víno! (velikost A5).

Obrázek B.2: Leták Chcu víno! - varianta 1 (velikostí vhodný do obálky).

(a) Varianta (a).



(b) Varianta (b).

Obrázek B.3: Leták Chcu víno! - varianta 2 (velikost A5).